University of the Pacific
Scholarly Commons

University of the Pacific Theses and Dissertations | Graduate School

2018

# Shape from focus image processing approach based 3D model construction of manufactured part

Mitchel Wendland
*University of the Pacific*, mwendland7@gmail.com

Follow this and additional works at: https://scholarlycommons.pacific.edu/uop_etds

 Part of the Mechanical Engineering Commons

## Recommended Citation

Wendland, Mitchel. (2018). *Shape from focus image processing approach based 3D model construction of manufactured part*. University of the Pacific, Thesis. https://scholarlycommons.pacific.edu/uop_etds/3569

This Thesis is brought to you for free and open access by the Graduate School at Scholarly Commons. It has been accepted for inclusion in University of the Pacific Theses and Dissertations by an authorized administrator of Scholarly Commons. For more information, please contact mgibney@pacific.edu.

# SHAPE FROM FOCUS IMAGE PROCESSING APPROACH BASED 3D MODEL CONSTRUCTION OF MANUFACTURED PART

by

Mitchel H. Wendland

A Thesis Submitted to the

Graduate School

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

School of Engineering and Computer Science
Mechanical Engineering

University of the Pacific
Stockton, California

2018

# SHAPE FROM FOCUS IMAGE PROCESSING APPROACH BASED 3D MODEL CONSTRUCTION OF MANUFACTURED PART

By

Mitchel H. Wendland

APPROVED BY:

Thesis Advisor: Jiancheng Liu, Ph.D.

Committee Member: JuEun Lee, Ph.D.

Department Chair: Chi-Wook Lee, Ph.D.

Dean of Graduate School: Thomas Naehr, Ph.D.

# DEDICATION

To Brittany who gave meaning to this effort.

# ACKNOWLEDGMENTS

My gratitude goes to Dr. Jiancheng Liu for his many hours of patient guidance while I worked through this project. I would also like to acknowledge the Pacific Fund for their sponsorship of my research. It would not have been possible without their financial help. I would like to thank Kevin Muller for his help and the use of his computer. He has been a great sounding board and a good source of advice and help when I needed it. I would like to thank Hua Ding for her help in taking image sets while I was away from the school. It would not have been possible to have run so many test cases without her help. I would like to thank my family and wife Brittany for keeping me focused on my goals and for all of their help and support through this whole ordeal. I couldn't have made it without you!

# Shape from Focus Image Processing Approach Based 3D Model Construction of Manufactured Part

## Abstract

By Mitchel H. Wendland

University of the Pacific
2018

The purpose of this research is to develop a process and an algorithm to create a 3D model of the surface a part. This is accomplished using a single camera and a CNC machine as a movable stage. A gradient based focus measure operator written in MATLAB is used to process the images and to generate the surface model. The scopes of this research are image processing and surface model generation as well as verifying part accuracy. The algorithm is able to create a rough surface model of a photographed part, and with careful calibration in a limited number of scenarios has been used in checking part z dimensions.

TABLE OF CONTENTS

# LIST OF FIGURES

Figure

# Chapter 1: Introduction

Image Processing and Machine Vision are some of the most recently emerging fields in Engineering. They make some bold promises of the capabilities they possess. Some have been explored, and others are waiting for researchers to discover them. The ability to measure a part without touching it, a robot that can sense distances, self-driving cars; all of these rely on machine vision, and image processing. Cameras can capture amazing details, but until recently we had to rely on human eyes to identify what was being photographed. This research is to develop a process and an algorithm to generate a surface model of a part using contrast as a focus measure.

The background for this project is predominantly in optics and image processing. The optical principles at work here are relatively well understood. The lynch pin for the project is the fact that a camera will have a definite focal length, and this length will be based on physical properties of the camera and lens combination. Camera and lens combinations will always have a definite focal length where objects will be sharpest or in best focus. The goal of this project is to exploit this physical property of optics to generate an accurate surface model of a photographed part.

Camera sees the world very differently than the human eyes in many ways, and in others it is very similar. They take light intensity and position on a sensor and convert that into data. This data is nothing more than a collection of ones and zeros within a computer until it is interpreted. The data is then interpreted by a variety of means and an image is generated from the data that will have some meaning to an observer. The field of machine vision is growing and

expanding very rapidly. The ability for a machine to sense something meaningful via optical means without touching the part is becoming highly sought after. Microscopes costing thousands to hundreds of thousands of dollars are being made and sold that can take critical measurements of parts and save the data for later review. These microscopes only need to have the part placed in a known location and then the machine can search for the measurements that it needs to take in pre-programed locations. Sometimes these use specific colors or shapes to verify part accuracy. With such an emphasis on the ability for a machine to see something it is no wonder why this field is growing so quickly.

The proposed method of photographing and then generating a 3D model of a part has some very interesting potential applications. If an accurate model of a part can be made, a computer could be used to measure parts quickly. This would represent a best case scenario if complete success is achieved. The ability to generate a model of a manufactured part or of raw material has a host of potential applications, and is one of the areas being researched in the field of machine vision.

The optical properties of thin lenses are well documented and provide the basis for understanding the proposed project. **Figure 1** shows a simple thin lens with focal length "f". This describes the way a lens refracts light rays. The focal length of the lens is an optical property. It is a known quantity and is constant. When light rays are parallel to the central axis of the lens the light will be focused to a point on the centerline and the focal length. This focusing property of lenses makes them ideal for photography.

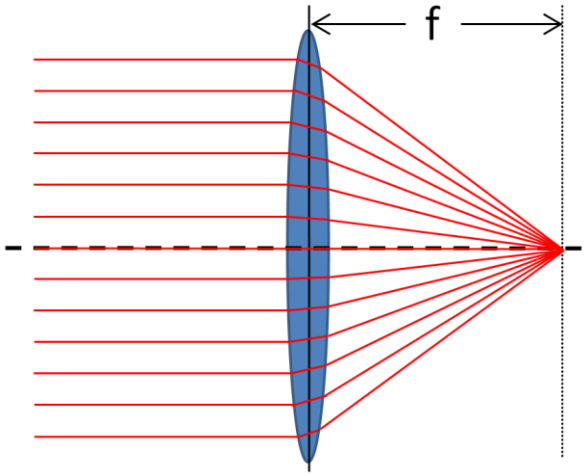



**Figure 1:** Simple Thin Lens with Focal Length "f"

In **Figure 2** a basic thin lens looking at an object and the real image that is generated can be seen as well as the thin lens equation. The thin lens equation describes the focal length of the lens based on the distance from the object to the lens and the distance to the real image of the object. This can be manipulated to mathematically predict real image location based on focal length and object distance. In order for the object to be in focus, the plane that the real image falls on has to be coincident with the plane of the sensor that is capturing the images. Otherwise the image will be out of focus and  blurry. The focus can be manipulated by the position of the lens and is in autofocus applications so that this is the case. Manual focus requires that the user adjust the position of the lens in order to bring the image into best focus.

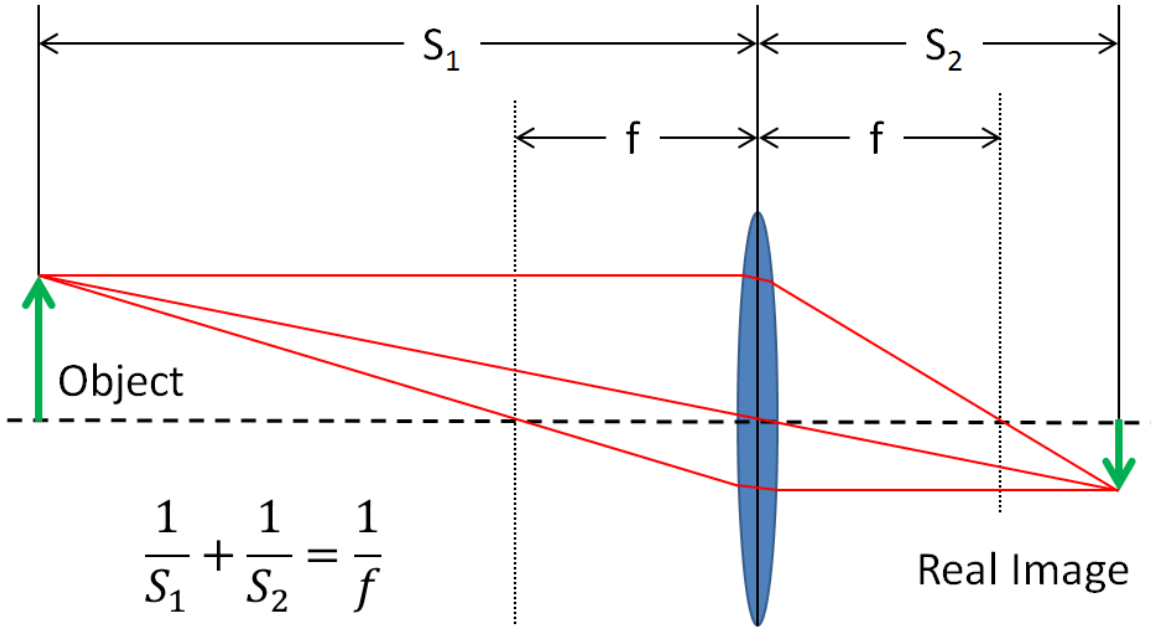



**Figure 2:** Thin Lens Equation Graphic

**Figure 3** shows graphically what the thin lens equation mathematically states. As the object moves closer to the focal length from infinity the real image moves away from the focal length towards infinity. Thus optical component selection is critical so that the distance that objects will be photographed at falls within the component combination's full range of depth of field so that sharp in focus images can be generated.

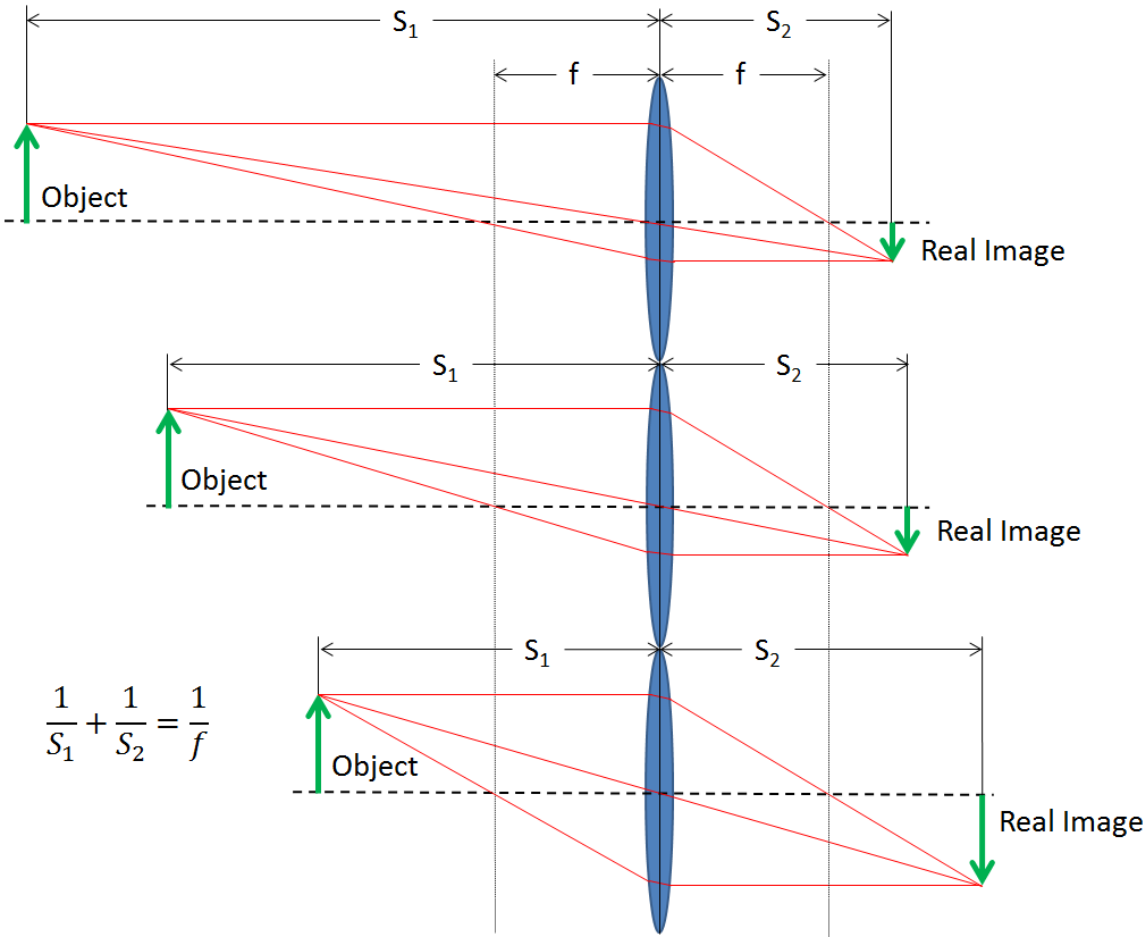



**Figure 3:** The Implications of the Thin Lens Equation

Images are stored in a computer with a variety of methods. MATLAB or Matrix Laboratory has several different methods it uses to handle images. The most common of these is to store it as an RGB image. This is a matrix of (M,N,3) dimensions with the dimensions of the image in pixels represented by “M” and “N”. The 3 represents the depth of the matrix required to store different values of red, green, and blue for each pixel. Another image storage standard is Hue Saturation Value or HSV. This stores the image in the same size matrix as an RGB image

but it stores hue, saturation, and value instead. The hue is the color of the pixel. The saturation is the colorfulness, or the depth to the color. The value is the luminance or brightness of the pixel.

Contrast is defined as the difference in color and brightness between one pixel and its neighbors, or in a grayscale image the luminance is the term that is used. The theory used in the development of the algorithm is that the contrast value will be at a peak when the pixel is in best focus. Based on the optical properties of the camera and lens combination, the location of that pixel when it is in best focus can be determined.

$$C_{i,j} = \frac{\sum_{m=-1}^{m=1} \sum_{n=-1}^{n=1} (L_{(i+n),(j+m)})}{L_{i,j}} - 8L_{i,j}$$

**Figure 4:** Contrast Equation

The formula used for computing contrast from each pixel to its neighbors can be seen in **Figure 4**. This equation is nested inside two for loops which steps it through every pixel in each image, except for the ones along each edge which have no neighbors along the outside. This particular configuration calculates contrast based on the luminance or value of each pixel in the image with the 8 pixels around it. The contrast values are averaged for each image. This can be used as an autofocus algorithm. The averaged contrast values are used to ensure that the images in the stack encompass the entirety of the object or scene. This allows a check of the image stack to make sure that the information needed for the shape from focus operation is present.

## Chapter 2: Review of the Literature

Focus Measure Operators (or FMO) are at the core of Shape from Focus or SFF, and numerous operators of various kinds have been proposed. The performance of a few has been studied in detail, and they can be sorted into several categories based on mechanism [1]. Those categories are; gradient based, Laplacian based, wavelet based, statistics based, discrete cosine transform based, and miscellaneous operators. Some focus measures have been proposed and discussed in more detail. The fast discrete curvelet transform was effective in enhancing the representation of sharp edges and discontinuities in a parts surface when paired with a bivariate shrinkage scheme for noise removal [2]. A novel SFF routine, based on combinatorial optimization, used a small subset of pixels called a neighborhood, and then computed the focus measure on the pixels in that neighborhood. The iterative process then tries to find the combination of those frames that result in the maximum focus measure for the pixels lying on those frames [3]. The effect of window or neighborhood size has been studied for its effects on the outcome of SFF [4]. An algorithm for 3D shape recovery is proposed in [5]. Principal Component Analysis is the technique of looking at a small neighborhood around each point of interest, and then uses regression analysis to create a depth map. Initially the depth maps that were created were extremely noisy, but the neighborhood based regression routine largely negates the effect of the noise while allowing for the creation of an accurate depth map.

It is important to know how accurate and reliable a depth map is and the first step to ensuring accuracy is by using the best quality images. A quality assessment for images with out of focus or blurred areas is proposed in [6]. This assigns a value to the out of focus areas on an image and then was compared to subjective evaluations of the quality of each image. It was

found to agree closely with human observations of an image. This would allow for a computer to determine if a particular image has the area of interest in focus or could be used as another focus measure for SFF.

Autofocus algorithms are used in cameras every day to make the pictures we take on our phones, and with our cameras better, and the end users don't need to worry about the subject of their photographs being blurry and out of focus. Thus they have been a subject of great interest since computer controlled lenses came into use. These autofocus algorithms are a useful tool to study in order to understand FMO in action. The difference between their use in general photography and in SFF is that in SFF our interest is in a particular pixel or small group of pixels instead of over the image as a whole. A focus measure for use in autofocus based on the medium frequency discrete cosine transform is discussed in [7]. Several focus measures were evaluated for use as an autofocus mechanism in a line scan camera [8]. The authors determined that the Tenengrad autofocus algorithm showed the most promise for their application. This method of autofocusing a camera computes the gradient of the image in X and Y by convolving it with the Sobel operators [1] [8]. Autofocus algorithms were tested for use on an optical microscope for a Mars lander [9]. An operator was needed that would quickly autofocus the microscope and also generate a single all in focus image from the series of images that the microscope took. It was determined that for all in focus image generation on this application gradient operators are most desirable. This seems to disagree with the evaluation done in [1] which determined that for the tested image stacks certain wavelet and Laplacian based FMO were superior. However, the team making the microscope autofocus algorithm was not attempting to perform a SFF operation; thus, this could account for their different conclusion as SFF is a different application. From these it can be seen that FMO and autofocus algorithms are specific in their application. Some

work very well in some applications and not as well in others. It is therefore important to select a focus measure operator carefully that will work in its intended application. For digital camera applications, an autofocus algorithm needs to be fast. A search algorithm is proposed to decrease time taken to find the best in focus plane that uses a hill climbing method to maximize focus [10]. An autofocus method using the discrete cosine transform is proposed in [11] for use in a micro vision system. The discrete cosine transform is used because typically a function can be represented accurately using less terms than a sine transform. This allows the algorithm to find the focal plane more quickly than other methods. An autofocus algorithm proposed in [12] and evaluated in [1] uses contrast in an image to calculate how in focus the image is.

Moving from the basics of FMO and autofocus algorithms, and on in to SFF. SFF has been a subject of interest for some time as the machine vision field has grown. A reliability measure for SFF is proposed in [13]. The algorithm analyzes the shape of the focus measure function and with no prior knowledge of the recovered scene identifies areas in the image stack that have low reliability and discards them. One research group proposed a SFF algorithm using a single image and a series of blurring and deblurring operations on the image to create an artificial image stack [14]. The method made no assumptions regarding the properties of the scene, and so didn't produce very accurate depth maps, which were only useful for observing general trends in the scenes topography. A method for SFF that works well in a noisy environment is proposed in [15]. It uses LULU operators and the discrete pulse transform to generate a 3D model of apart. The LULU operators are particularly effective in removing noise from a dataset, are non-linear, low complexity, and very efficient. It is a clever noise reduction operation, and it has two parts. The lower operator takes the data points around the target point and forms a mini sequence of the values of the points around each of them. It then takes the

minimum of each of the mini sequences, takes the maximum of the results of that operation, and afterwards defines the data point as that value. The upper operator does the same but with the min and max functions swapped.

SFF has numerous uses in many fields, and applications for SFF are of great interest to processes which rely on optical inspection to verify compliance with requirements. An autofocus was used to determine if the stage where the inspection was to take place was sloped, and his data was fed back to the computer controlled camera reducing the number of images that need to be taken in an inspection session to adequately determine compliance [16]. This application depending on factors such as lighting and part surface texture could be ideal for SFF.

Another common focus measure operator's task is the generation of an all in focus image and, in SFF applications, overlay that onto the generated depth map. This is typically done by taking areas or individual pixels from the image stack that are determined to be in best focus and stitching an image together based on all of the areas or points in best focus. This is of particular use in the field of microscopy where depth of field is typically very shallow and the ability to look at more of if not the whole picture is tremendously useful. A method for doing this on board a digital camera is proposed in [17]. A surface area based method for creating an all in focus images is proposed in [18]. An all in focus image was created with only two initial images: one with the subject in focus, and another with the background or a second subject in focus. This drastically cuts down on the number of images needed and speeds up the generation of the all in focus image considerably.

A specialized application of SFF is in scanning laser microscopy. Lasers have been used in rangefinders for some time, and they have also seen use in microscopes for 3D scene

reconstruction in the past few years. The systems that use lasers for long distance range finding use a timing system and sensors that detect specific wavelengths to determine distance to a target very accurately on a large scale. Some laser microscopes pass a band of light with a very precise and known focal length over the scene to be reconstructed and, as this beam sweeps over the surface, the laser's focal length sweeps in or out. ] Thus, by finding the spot of greatest intensity and knowing the optical properties of the laser's scene, the depth map can be reconstructed with a high degree of accuracy. This method for distance finding was tested over  distance up to 1 meter and was found to be effective [19]. This system required a relatively simple camera with no moving parts to be effective; this stands in contrast to other methods that require a computer controlled lens for the camera. It should be noted that the equipment to facilitate a sweeping laser with variable focus distance is quite expensive. This method offers the benefit of working on scenes with poor texturing that other FMO find difficulty in recovering accurately. The accuracy of a coherent illumination source verses a focused standard light source was tested, and the laser was determined to be the superior illumination method. Even with a simple camera with no moving parts, this system still qualifies as SFF as it relies on the focusing of a beam of light.

# Chapter 3: Methodology

The processes to generate the surface model fall into two main categories; taking the images, and processing the images.

## Procedure

The image processing step is very sensitive to how the images are taken. Thus the process used to take the images can be viewed as the foundation for the rest of the research. The camera used is a Basler Ace acA645-100um monochrome USB 3.0 camera with a 4.5mm compact fixed focal length lens from Edmunds Optics. The camera and lens is held in the CNC machine via a custom designed 3D printed mount which is held in in a 3/8” tool holder. The camera attached to the mount can be seen in **Figure 5**.


**Figure 5:** Camera on CNC Mount

**Figure 6:** Test Part.

The part was 3D printed on a Makerbot Z18 printer, is 1” by 1” by 0.5” overall, has a variety of features on its surface to test the algorithm’s sensitivity, and it can be seen in **Figure 6.** It was made on the highest accuracy that the printer could support. The 3D printed part offered a textured surface that was easier for the algorithm to see than a smoother machined part. The part is placed in the vise in the CNC machine, and it is surrounded with matte grey lightly textured paper that always looks out of focus to the camera. This provides a neutral and non-reflective background so the algorithm can focus exclusively on the part and not on the reflected lighting in the background. If needed, matte black masking tape can be used to hold the paper in place. However, care should be taken as the masking tape, although matte compared to other types of tape, is far shinier than the paper and will show up to the image processing algorithm.

The lighting ring is placed on a set of parallels for the CNC vise so that it is effectively illuminating the part and does not move with the CNC machine as the images are taken. This setup can be seen in **Figure** 7. It is important for the lighting to be consistent from picture to picture as any variation in lighting will ruin the algorithms ability to generate an accurate 3D model. The amount of lighting was not extensively tested. The lighting was simply adjusted until it offered the best contrast on the surfaces of the part to be measured checked by a live stream to a computer display. The amount of lighting and type is an opportunity for further study. Once the lighting was in place, the camera was maneuvered into position in the center of the lighting ring and also over the center of the part. The cameras aperture, focus, and position was adjusted so a sweep of images can be taken. The easiest way to do this is to adjust the position of the camera to about 2” over the part, and then to adjust the focus so the bulk of the part is in focus verified visually on the computer. Then the aperture should be set as wide as possible so the depth of

field is as shallow as possible. The lighting should be readjusted so the contrast is as high as it can be in the image.


**Figure 7:** Experimental Setup in CNC Machine

Then move the camera in the z- axis downward towards the part through the intended span of the image sweep. The step size used was .01". This provided a good balance of accuracy and use of time as the CNC machine's position had to be manually adjusted between each image. The images were saved as a numbered sequence so they would be easy to load in order into MATLAB. If taken correctly the images start with the object smaller and out of focus and the object will grow progressively larger and more in focus. The focus should peak. Then the object should continue to grow in relative size but get further out of focus. The opposite could also work, however the algorithm is set up to only work in the direction specified.

Once the images are taken it is important to verify that they have been taken correctly. The numeric sequence must be perfect with no gaps, and each image has to be correct. That is to say that the object is always growing progressively larger as the sequence goes along. If not, either the process needs to be restarted or the images missing from the sequence will need to be taken. It is best to verify the images before taking any of the experimental apparatus down so that images missing from the sequence can be easily retaken without having to set up the whole apparatus and redoing the entire image stack. The first and last images in the stack can be seen in **Figure 8** and **Figure 9.** The image series will be referred to from here on out as a stack because the images focal planes amount to this if visualized in 3D. The focal plane has the same x and y dimensions in each, and since the camera moves in the z direction, the area with the position information can be visualized as a rectangle. Because of lens distortion this is not technically the case; however, the distortion is small enough to be negligible compared to the size of the image.

The first major hurdle for the code is to deal with the size change of the object. Others have dealt with this by doing the focus sweep with the lens instead of moving the camera. As a lens with a computer controlled focus was more than a little out of the budget and a CNC

machine was readily available, the computationally more difficult method of moving the camera and correcting the size change with software was selected.

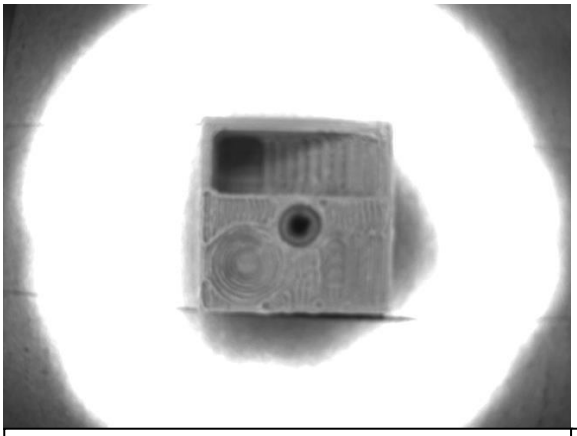


**Figure 8:** First Image

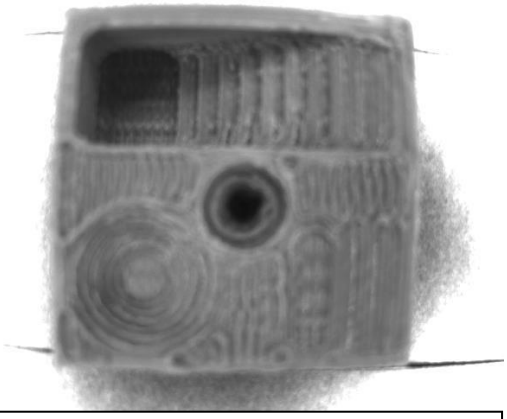


**Figure 9:** Last Image

The code loads and crops an image to the same relative size as the final image in the stack, but the image retains the same number of pixels and the first images cropped size. The images further along in the stack have progressively less and less cropped out of them and are then resized to the same number of pixels as the first image in the stack. This allows the code to compare the same pixel across the whole stack to the ones immediately before and after it in the sequence. This becomes very important when comparing image contrast values from one image to the next.

As they are loaded, the images are converted from a RGB image to a HSV image. This makes the contrast calculation faster, especially since the images are grayscale. The speed increase is due to the way luminance or brightness is calculated on and RGB image versus an

HSV image. On the RGB image, all three values are needed for the calculation. For an HSV image only one value is needed for a contrast calculation. The images are then cropped and resized. Then the contrast is calculated for each image. It is done by comparing the Value value of each pixel to its neighbors. The contrast calculation's definition of neighbors can vary based on a specific input. The best contrast calculation was achieved with a plus shaped definition of neighbors or, if computation time is less important, all 8 of each pixel's adjacent neighbors can be used. These values are saved in a matrix where the x and y values are the contrast numbers, and the z coordinate represents the number of the image in the stack. The contrast matrix is taken and then the z values for each pixel are determined. The maximum value for each xy coordinate as z is stepped through is saved and assigned a set of position coordinates determined by the step number, step size, and initial position information. A 3D depth map of the part can then be generated. It was helpful to overlay one of the central images on the depth map to more easily visualize the part as the computer sees it.

This depth map for the photographed part was very noisy and left a lot to be desired. A noise filtering and smoothing algorithm was developed to make the 3D model more closely resemble the real part. Noise filtering deleted a lot of the noise, and as more passes through the filter were run, the noisy original began to quite closely resemble the parts actual dimensions. The surface still appears bumpy, but on average and under very specific circumstances the algorithm was able to detect the parts dimensions with a reasonable degree of accuracy.

**Analysis**

Many methods were explored to get to the final process discussed above; both for the taking and processing of the images. Taking the images required rather less time to achieve than in processing them correctly. The images need to be taken so that they have a known gap in

between all of them. A uniform gap was selected in order to make the program easier to run; however, a non-uniform image gap or step size could be used as long as it was noted and compensated for in the code.

The background was selected after initial attempts at taking the images produced very poor pictures. As the algorithm picks up very well on contrast the lighting on a metallic part or background tends to be a very easy thing to pick up. This leads to a very inaccurate depth map that contains a lot of bad information. The metallic background in the CNC machine especially with the initially selected lighting made the first batch of pictures completely unusable. Therefore, the matte background was selected in order to make it easier to see the part.

The lighting is another critical aspect of this research. At first the lighting was just ambient light from the room; this proved to be inadequate. A ring light was selected next and was initially attached so that it moved with the camera. This caused a dynamic shadow environment, and it led to another bad batch of photographs. Finally the ring light was set on a pair of vise parallels so that the camera could move independently of the lighting. This gave the most consistent pictures, and led to the only usable or even reasonably accurate depth maps.

The parts construction also had a great deal to do with the usability and accuracy of the depth map. A quarter was also photographer in the same conditions as the 3D printed part. The metallic surface of the quarter caused the depth map to be very inaccurate. The relatively rough and more matte appearance of the surface of the part from the 3D printer offered a much better surface to the algorithm.

The contrast part of the calculation is actually one of the easiest things that had to be done to make the code work. The contrast calculation was one of the first things that were

written for the code. An algorithm was made to calculate and display a figure of the contrast for a single image so it could be fine-tuned for the application in isolation. It was initially noticed that the code would pick up sharp edges very well. The sharper the contrast the better the code would pick up on features in the image. This should have been obvious as the whole point of that piece of code was to calculate contrast. It was a very different way to look at an image. The implications of the way the computer sees images were not immediately apparent.

Several different ways of calculating contrast were tried. Eventually the 8 point and plus methods were selected to be included in the final version of the code. The 8 point was the most sensitive method that was tried. It gave the best contrast maps of the images. The + configuration was very close, as was the X configuration. The X was slightly less sensitive than the +, but the difference was almost negligible. The reason the + configuration was left in was to speed up computation time if many trials were needed later on as it cut the number of operations by quite a lot. Once the method of contrast calculation was selected and fine-tuned then work began on the rest of the code.

The next difficult step was to compare pixels from one image to the next. It sounds straightforward, but this represented the most significant challenge of the entire project. The problem arises as a result of the need to have the whole part in focus in the image stack and how the image stack is physically generated. The image stack has to be generated by moving the focal plane through the entirety of the part. This can be accomplished by a variety of means. The easiest and probably best way is to use a camera that has a computer controlled or mechanically controlled and very accurate focal distance. The other way is to use a camera with a fixed focal length and then to move the camera. The second method was selected due to budget constraints and proved to be much more difficult from a coding standpoint. This is because the object's

relative size changes as the camera moves closer and further from the part. This was overcome in this case with modification to the code. The first image and the last image are compared. The relative size change of the object will be a function of the distance that the camera moved and the distance from the camera to the part. Once the size change is known, the last image taken where the camera is closest and the part is largest is compared to the size of the first image and the first image is cropped to size. This is repeated for the rest of the images so the object is the same size in each image. Each image has a different number of pixels, and the code is not robust enough to handle a change in the number of pixels from image to image as it needs to compare one pixel in one image to the same pixel in the next image. To take care of this, the images are all resized to the same number of pixels that the first cropped image has. It could be interpolated and could go the other way, but it is not best practice to create more information from less. All of the images are now the same size pixel wise and the object size is the same in each image, so the pixels can be compared to their counterparts after the contrast is calculated. This was tried several different ways before the final iteration was settled on. First a correction factor was implemented, it didn't work at all. It attempted to change the size of each image based on its position. The next idea was the one that eventually worked. It took several iterations, but the end result was the cropping and resizing algorithm currently in the code.

The issue that this doesn't solve is that of image distortion. The images are cropped and no attempt is made at minimizing distortion either radial or tangential. The calibration of the camera to this level of accuracy was not accomplished, and thus the images retain a level of distortion which contributes to some of the inaccuracies seen in the depth maps. This could have been overcome with additional software, but for the purposes of this research as a proof of concept it was not necessary, and would have taken much more time. This distortion will be

different in each of the images after cropping because the camera is moving. This would at least be the same in each image if the focus sweep was done without moving the camera.

# Chapter 4: Results

## 3-D Printed Part

If sufficient care is taken when gathering the images, the focus measure operator is able to create a 3D model of the photographed part. The first and last images contrast values can be seen plotted in **Figure 10** and **Figure 11**. First the contrast calculation is completed on each individual image. The generated initial model is often very noisy as can be seen in **Figure 12** which is a raw depth map as the computer sees the part. The depth map with the central image overlaid can be seen in **Figure 13**. The noise is more apparent in the second of those figures. By looking at these alone, it is difficult to see if the data supports any conclusions due to the apparent level of noise. However with a smoothing filter applied, the 3D model begins to more closely resemble the part as shown in **Figure 14**. **Figure 15** shows the generated model with the photograph overlaid. The smoothing filter tends to round off any sharp corners, so depth measurements need to be taken sufficiently far from edges to avoid the rounding effects of the filter. In addition, the filter pads the edges of the images with zero values. This is why the edges appear to sharply turn up all the way around, and they are trimmed off for some of the plots. This effect makes measurements near the edges impossible.

Once the code has been calibrated, the objects size and shape can be estimated. **Figure 16** and **Figure 17** show the part's depth map to scale with the middle image overlaid and with a grid overlaid. This can give some idea what the part looks like. The accuracy of the depth map at a few locations is appraised in **Table 1**. These locations can be seen in **Figure 18**. The areas circled in red are the locations that the height was compared to the surface level which is circled

in blue. The depth map is accurate in some areas and much less so in others. A contour plot in **Figure 19** shows the relatively flat surface of the block. **Figure 20** is a plot of the average contrast value in each image as going through the stack. From this the image can be estimated e when the part is best in focus as the peak on the curve. This logically follows our expectation for the images to start out of focus and then to go more in focus as the object's visible surfaces pass through the focal plane. Then as expected the focal plane passes through the objects surface the level of contrast declines.

| Data Point | Block Surface Level | Pocket Depth | Inner Circle Depth | Wide Channel Depth |
|---|---|---|---|---|
| 1 | -0.5072 | -0.6952 | -0.5515 | -0.5912 |
| 2 | -0.5142 | -0.7363 | -0.5578 | -0.6111 |
| 3 | -0.5027 | -0.6875 | -0.5672 | -0.5954 |
| 4 | -0.5275 | -0.7296 | -0.5575 | -0.5779 |
| 5 | -0.528 | -0.6993 | -0.5414 | -0.5608 |
| 6 | -0.5034 | -0.7309 | -0.5662 | -0.612 |
| 7 | -0.5052 | -0.7007 | -0.5607 | -0.5573 |
| 8 | -0.5176 | -0.6993 | -0.5686 | -0.6015 |
| 9 | -0.5026 | -0.7415 | -0.579 | -0.566 |
| 10 | -0.5079 | -0.7315 | -0.559 | -0.6087 |
| Averages: | -0.512 | -0.715 | -0.561 | -0.588 |
| Height Difference | | 0.204 | 0.049 | 0.077 |
| Actual Height Difference | | 0.2 | 0.04 | 0.004 |
| Percent Difference | | 1.77% | 23.15% | 1814.00% |

**Table 1:** Generated Model Accuracy Appraisal

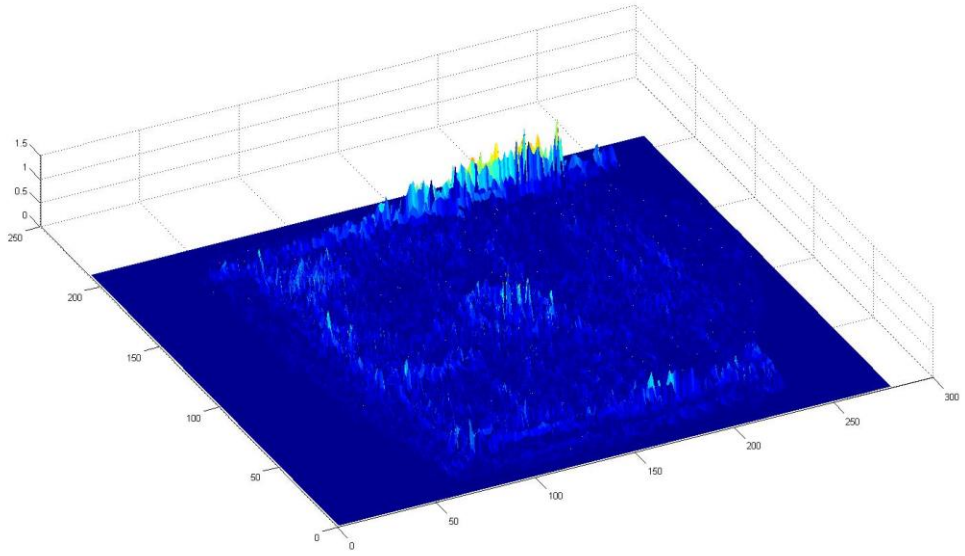



**Figure 10:** Contrast Values of the First Image

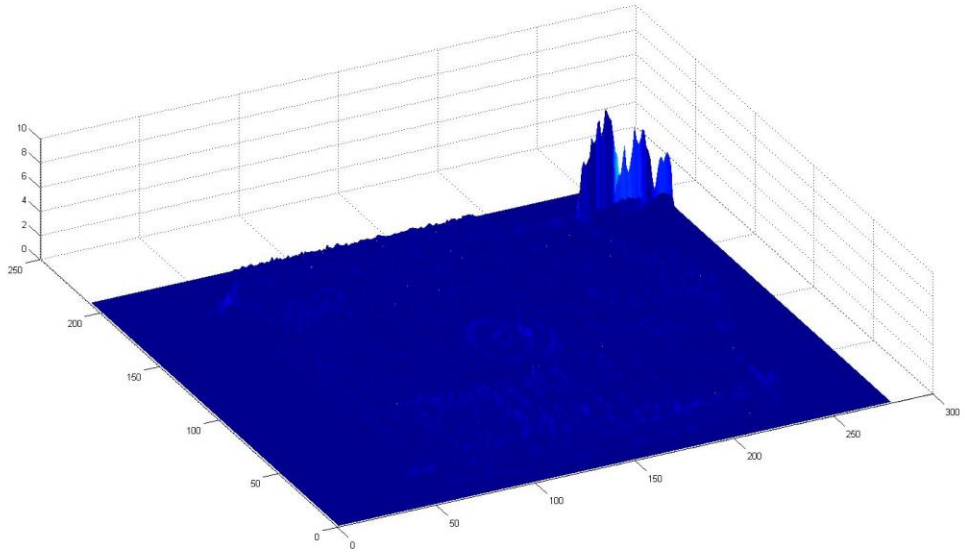



**Figure 11:** Contrast Values of the Last Image

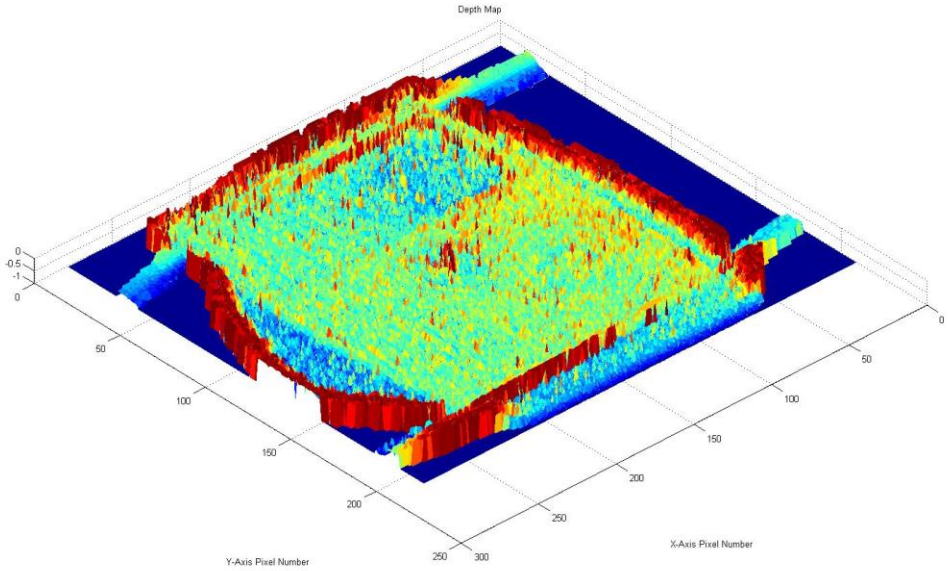



**Figure 12:** Raw Depth Map not to Scale

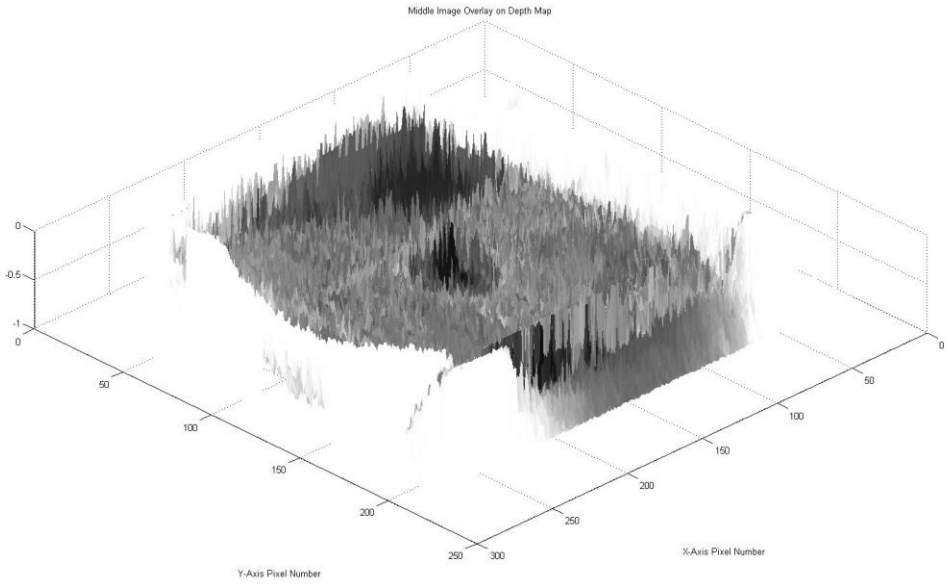



**Figure 13:** Depth Map with Middle Image Overlaid not to Scale

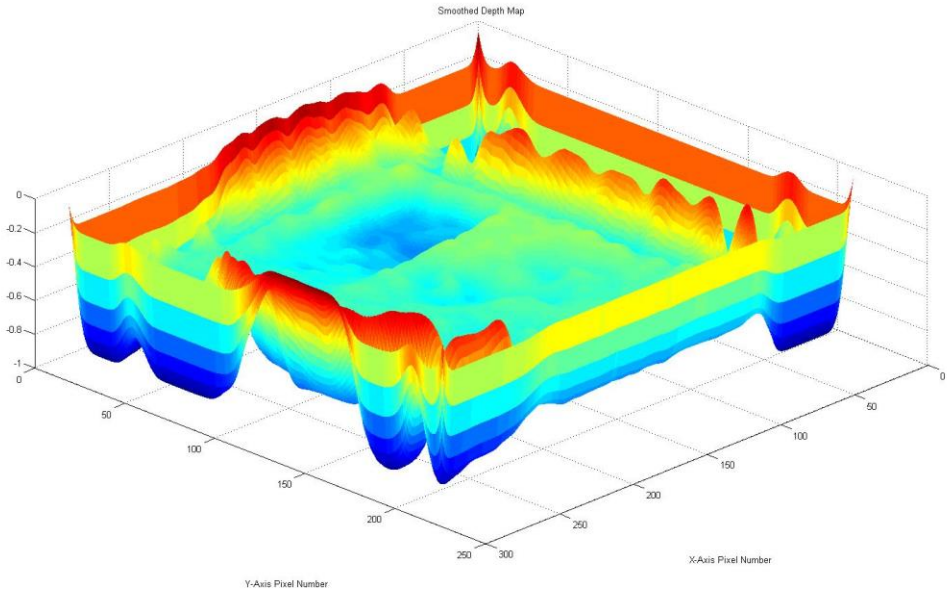



**Figure 14:** Smoothed Depth Map not to Scale

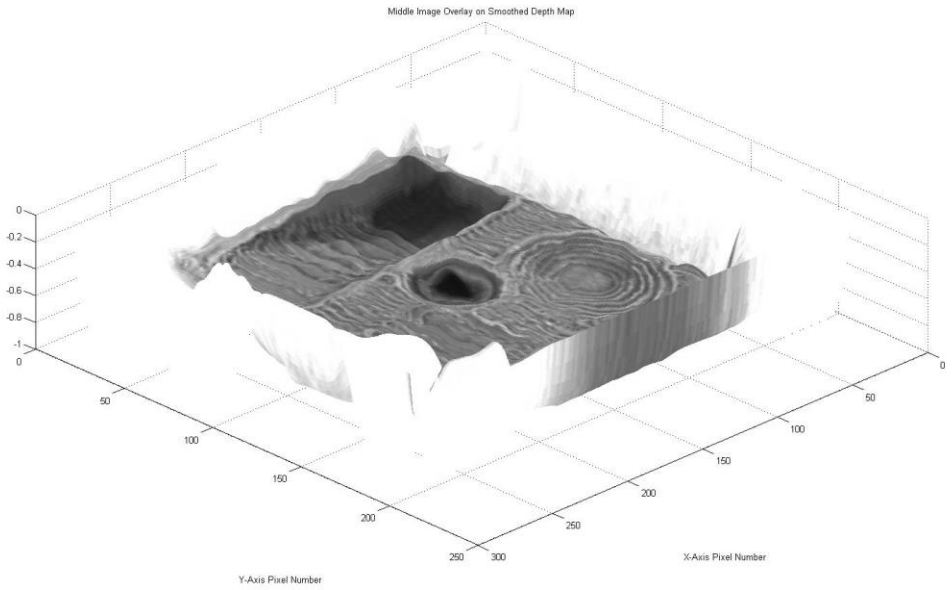



**Figure 15:** Smoothed Depth Map with Middle Image Overlaid not to Scale

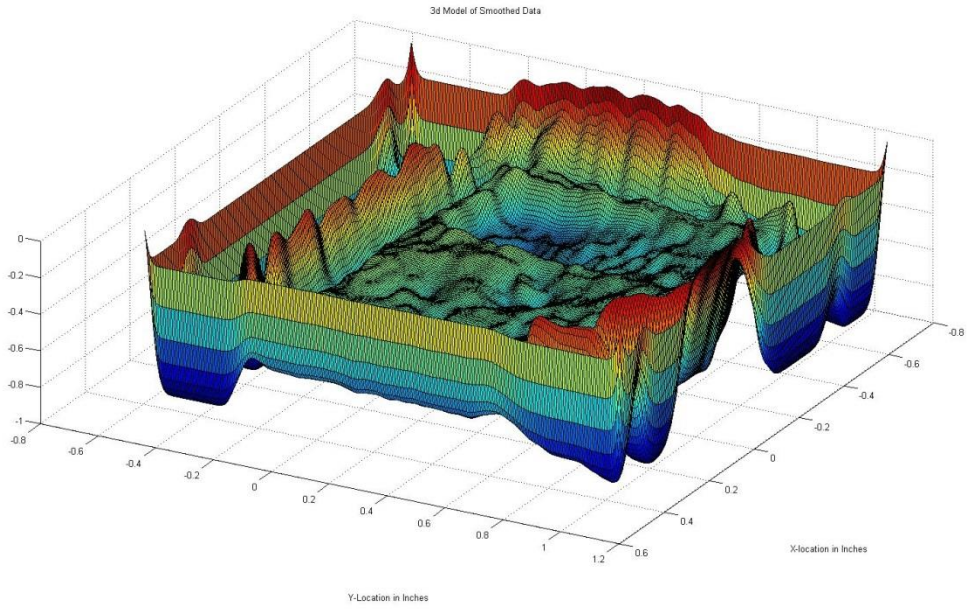



**Figure 16:** Generated Model to Scale

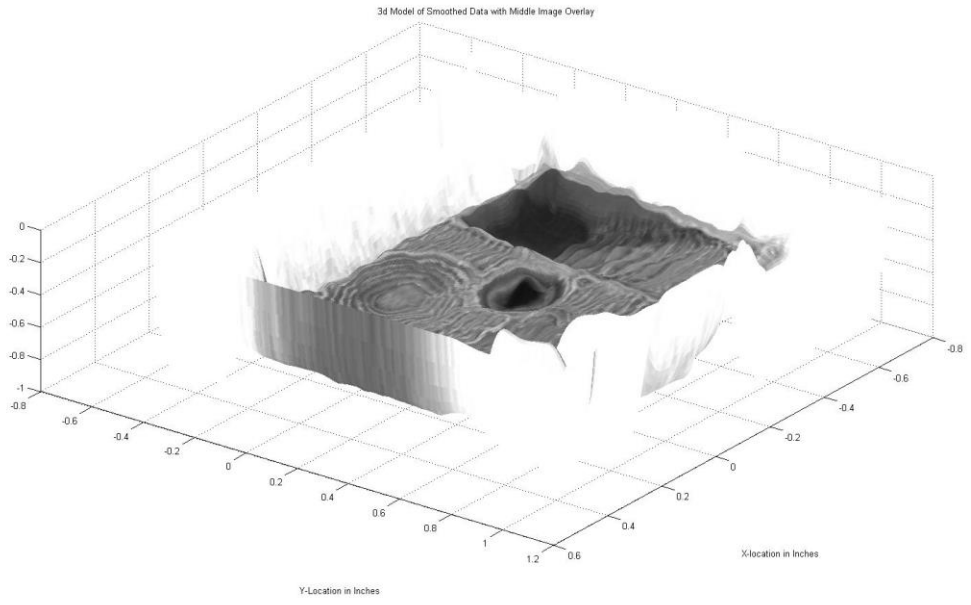



**Figure 17:** Generated Model to Scale with Image Overlaid

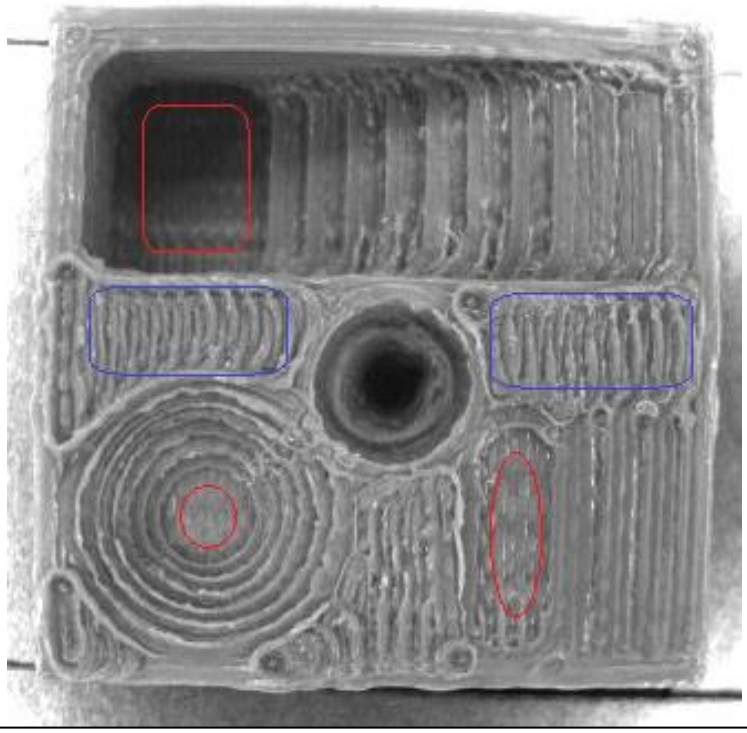


**Figure 18:** Sample areas: Surface Height Zero Level Taken inside Blue Perimeters, Areas Compared to Known Values in Red.

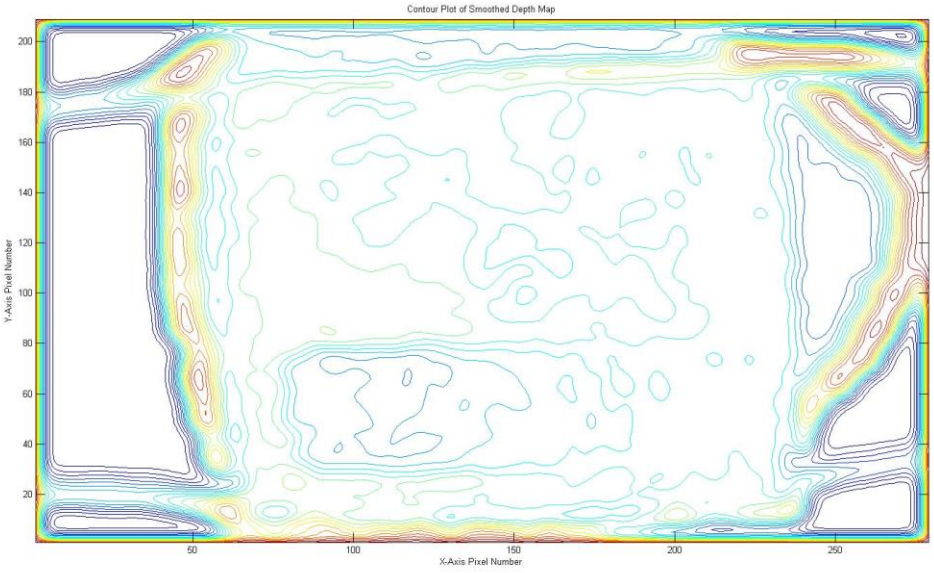



**Figure 19:** Contour Plot of the Top Surface of the Part

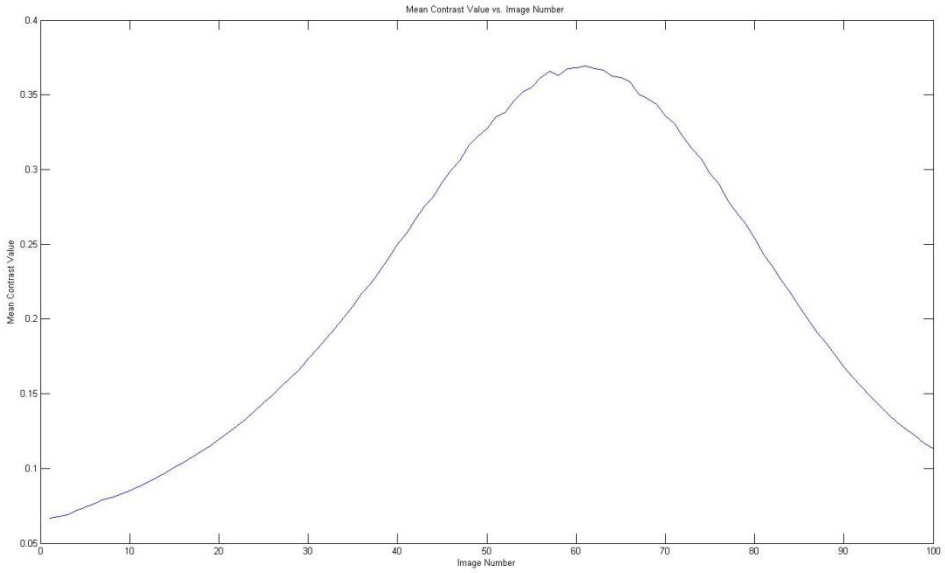



**Figure 20:** Mean Contrast Value in each Image vs. Image Height

**Quarter**

An accurate depth map of a US quarter was also attempted as an exercise to see how well curved, smooth, metallic surfaces would appear once processed. This did not result in an accurate depth map, but it did show some of the limitations of the algorithm and image gathering techniques. The depth map generated from the resultant images is far from looking like the actual quarter. The outline can be seen fairly easily, as well as the outline of President Washington's head. Any detail beyond that is nonexistent. The features are too close together in the z-axis to be accurately measured especially as smooth as they are. The metal surfaces reflective properties also make producing an accurate model difficult. More care was taken when setting up the lighting for this trial, and the map was much better than expected. No measurements were taken on the depth map for the quarter. It was far too inaccurate to produce any meaningful results. It

served only as a technical exercise to see how the algorithm would do on an object with a different surface finish and different features.

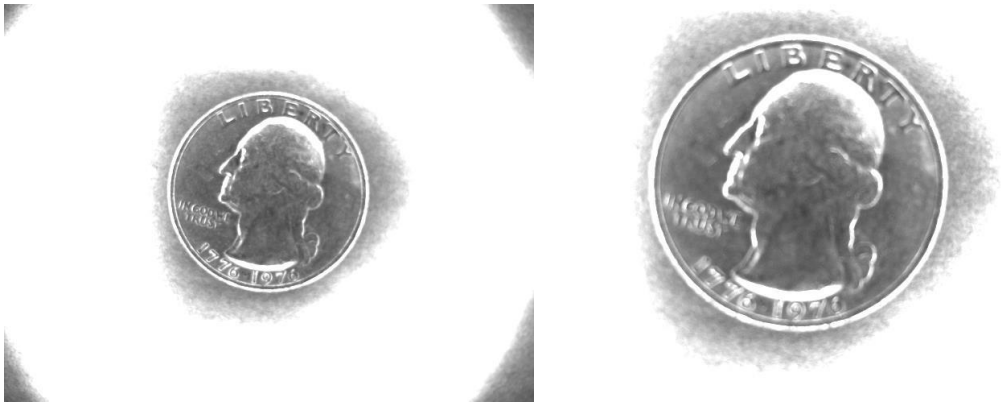



**Figure 21:** First and Last Images of the Quarter

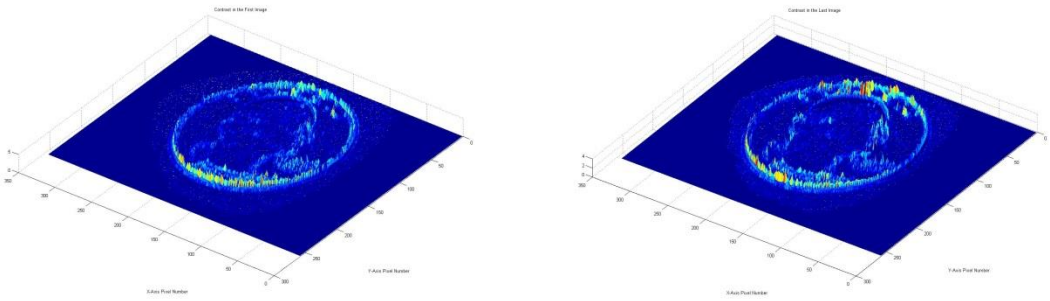


**Figure 22:** First and Last Images of the Quarter after Contrast Calculation

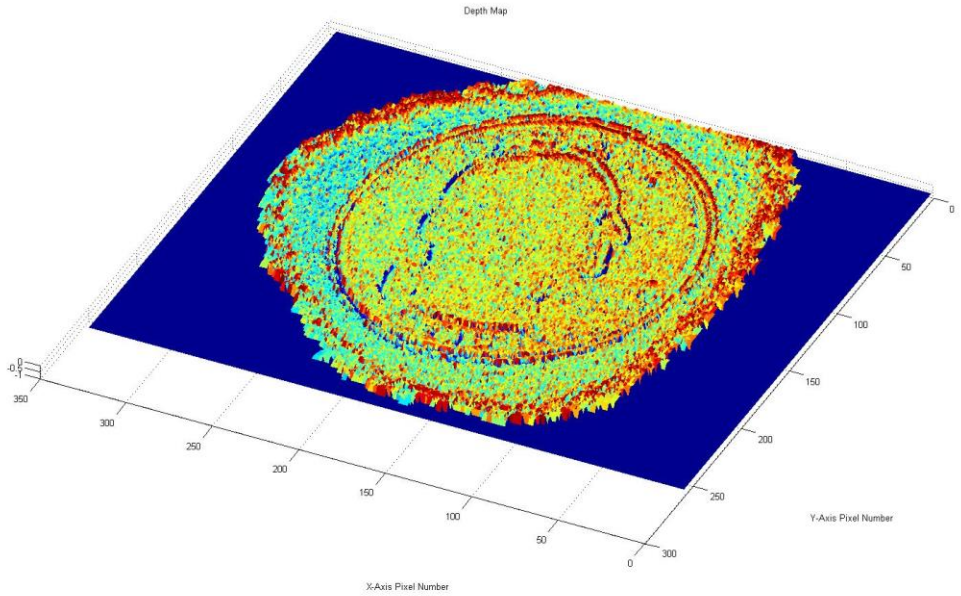



**Figure 23:** Raw Depth Map

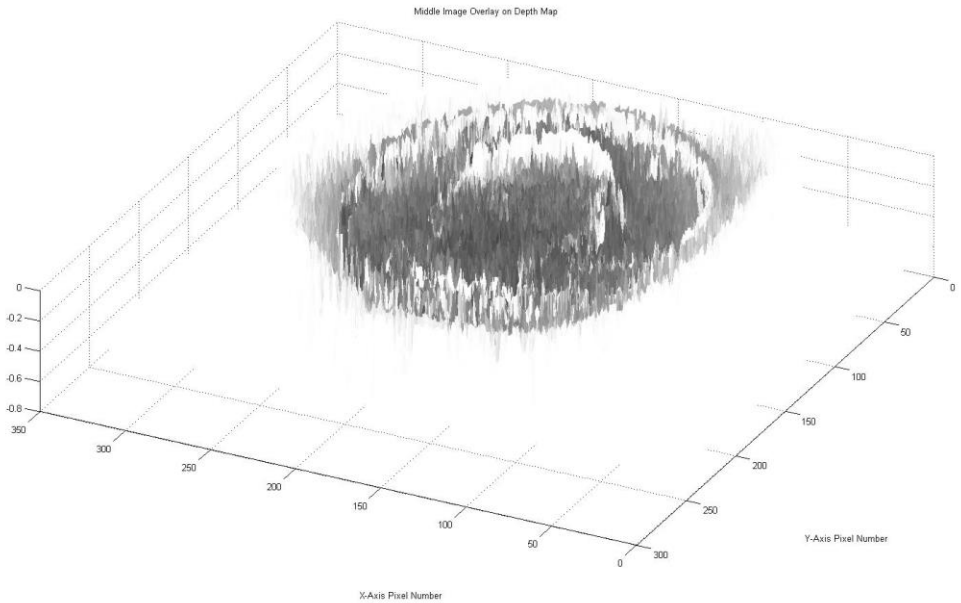



**Figure 24:** Depth map with Image Overlay

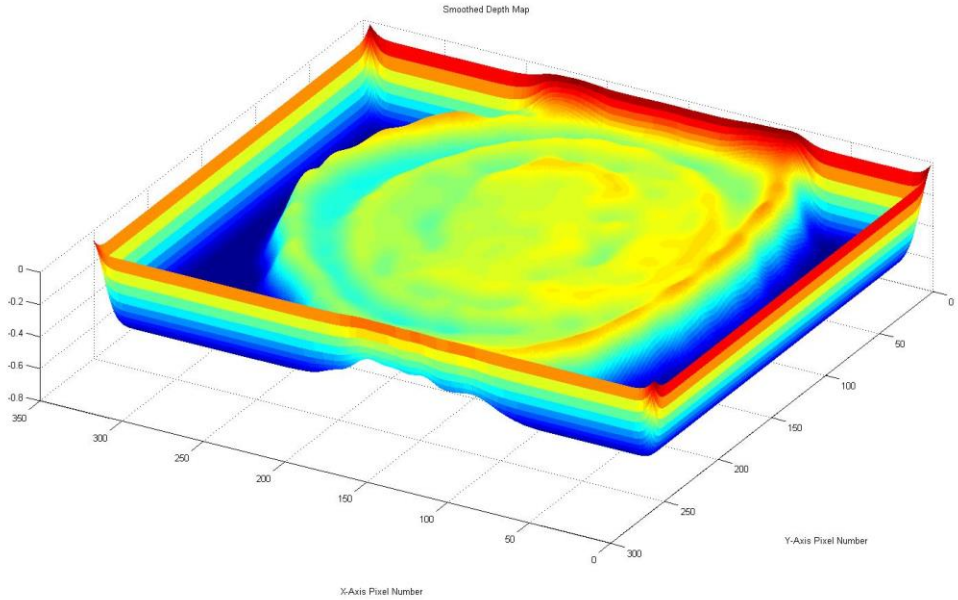



**Figure 25:** Smoothed Depth Map

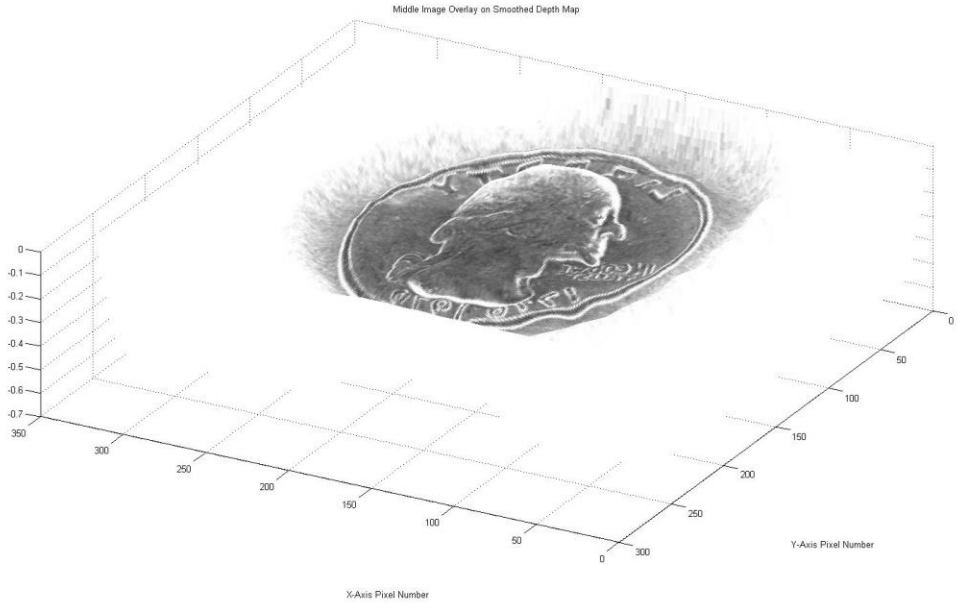



**Figure 26:** Smoothed Depth Map with Image Overlay

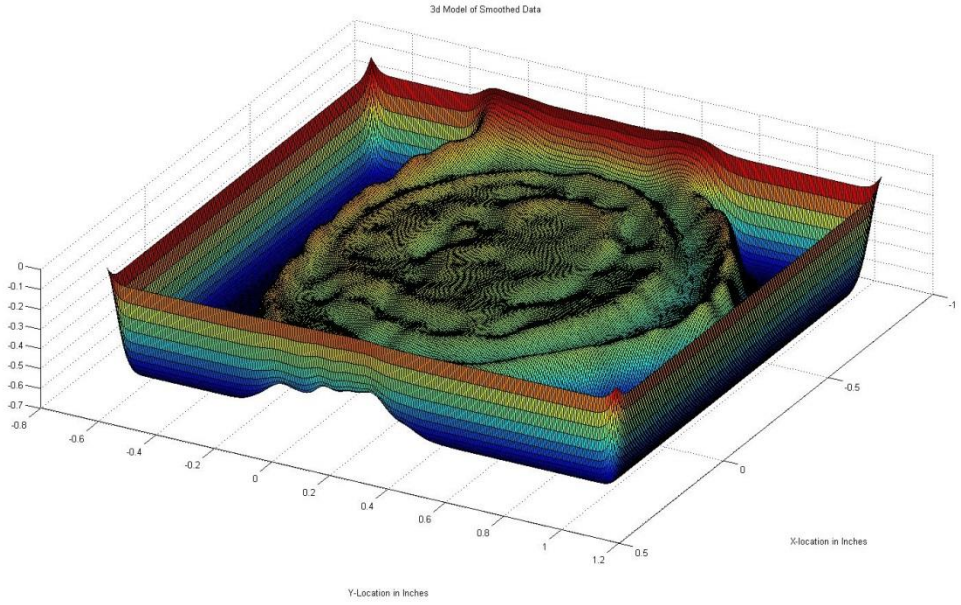



**Figure 27:** Generated Model to Scale

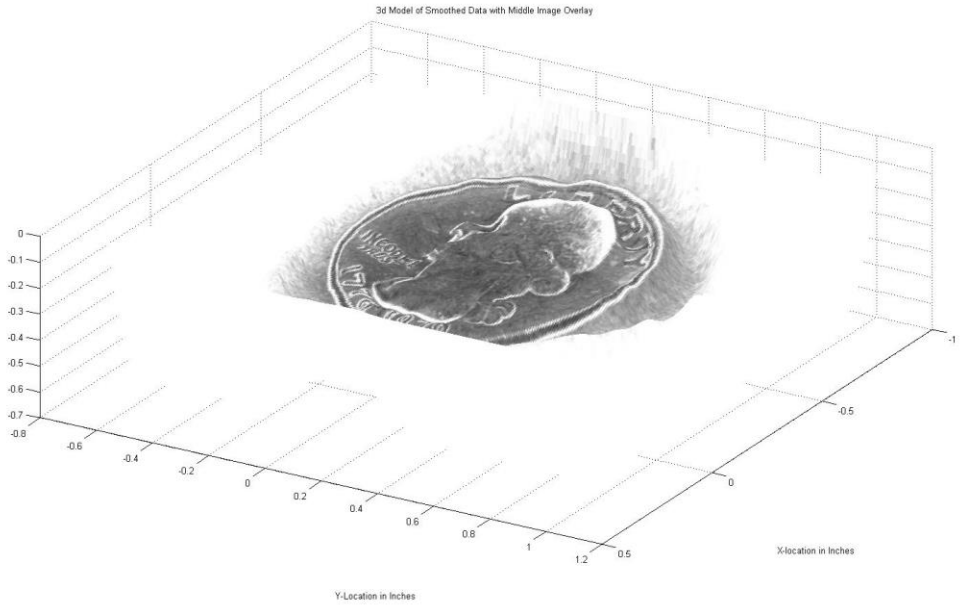



**Figure 28:** Generated Model with Image Overlay to Scale

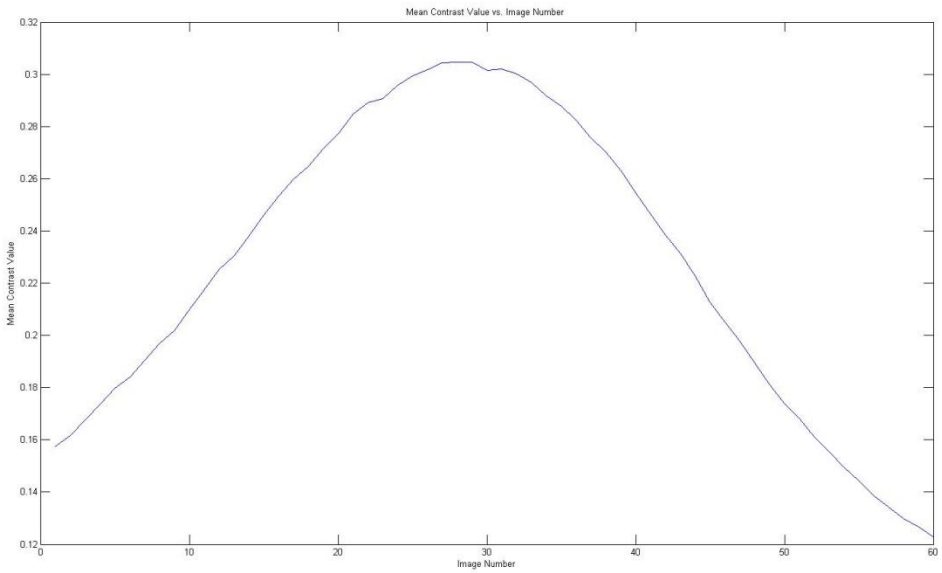



**Figure 29:** Contrast vs. Image Number for the Quarter

A total of 12 objects were photographed to test out the algorithm on a variety of materials and textures. The results were inconsistent. **Figure 30** shows the resulting surface contour plots. The plots are a contour plot superimposed over the 3D surface plot. This allows for slightly easier visualization of the parts surface as seen by the camera. The models which most closely resemble the shape of the photographed parts have one thing in common. They all have a very rough surface finishes and, in this case, are fracture planes on steel test samples. This would support the conclusion that surface finish is of the utmost importance when attempting to get accurate results; however, fractured surfaces do not lend themselves to making measurements to compare to the models. Parts with a machined finish or a polished finish do not display sufficient

contrast to make for accurate plots. The parts that were 3D printed, one of which was the first sample discussed above, generally made for good surface plots.

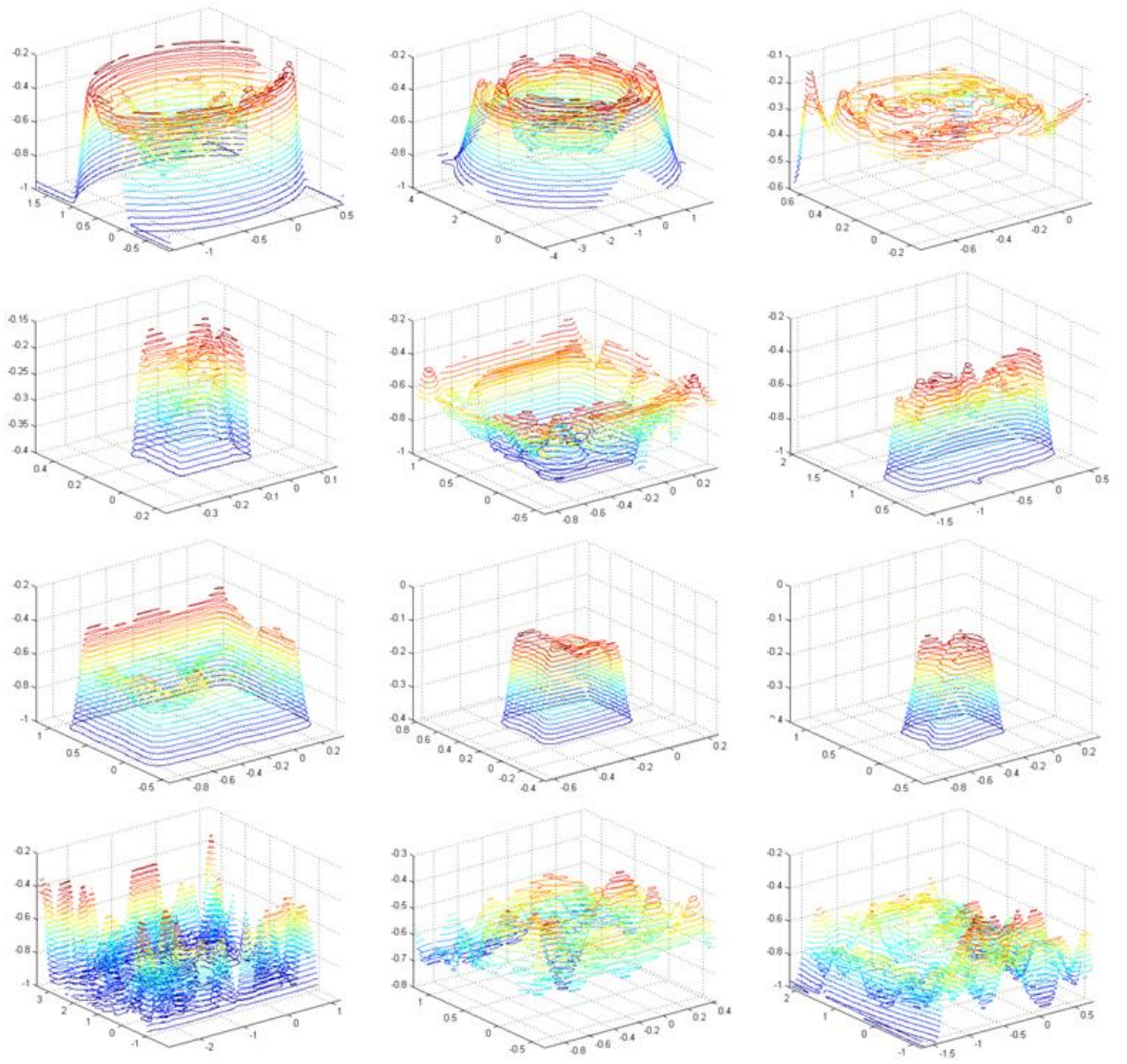


**Figure 30:** Resulting Surface Contour Plots

# Chapter 5: Discussion

The algorithm works reasonably well in carefully controlled circumstances. It can detect pockets and their depths fairly well as long as the pockets are sufficiently large and deep. The created 3D model is a reasonable representation of the part. The accuracy for smaller shallow features is nonexistent. They are lost in the noise and do not reliably show up in the depth map. This algorithm exceeded my initial expectations. It appeared that focus measure would not work well in this application, but as long as the data is smoothed sufficiently, and the conditions are just right, the depth map is reasonably accurate for larger features. The lighting requirements as well as the other environmental requirements make this a finicky process; unless the environment is controlled very carefully this process will not work. The lighting, the texture of the part, the background, and the calibration are critical. Even with so many factors have an impact on the results, it was surprising that it worked as well as it did.

It is important to note that for the best accuracy, as far as Z level measurements are concerned, it is recommended not to measure near the edges of the image and the edges of the part. The features that are measured also need to be larger than the step size used when taking the images. The accuracy could be improved by the use of a smaller step size; however that would only be practical if the process was sufficiently automated. Another method for increasing the accuracy could be to change the way of calculating the maximum contrast. As it is currently implemented, the code selects the maximum value from an image at a discrete z value thus tying the accuracy to the step size. A more sophisticated method for selecting the contrast maximum would involve creating a function for each of the pixels and then finding the maximum value of that function. This would allow for values other than the discrete values for each image in cases

where the maximum contrast would lie between images and shows potential for increasing the accuracy of the algorithm. The smoothing algorithm tends to smooth out any sharp edges that would be present in an actual part. It also warps the edges of the image, so no measurements can be taken there. The accuracy for measurements in the X and Y directions was not tested. The distortion of the lens and sensor combination would have been difficult to correct for by developing something from scratch. An algorithm to correct images for lens distortion exists in the MATLAB Image Processing Toolbox. However that particular tool is only available for R2014 and later MATLAB installations. This project was coded on version R2010a, so no such built in tool exists.

Image sharpening within MATLAB was tested for artificially increasing the contrast in the images. This gave mixed results. One pass through an image sharpening algorithm produced images with higher average contrast but no apparent increase in accuracy in Z level measurements. It also made the plots considerably noisier so it was abandoned. Another idea that was tried and quickly dismissed was interpolation of the initial images to increase the number of pixels. It would not have increased the accuracy of the plots and considerably increased time taken for computations to be complete. One image took almost 16 seconds just to do the contrast calculation after the interpolation so it too was left out of the final iteration of the code.

The algorithm and process have some definite limitations. It only works if lighting is done very carefully and in a very specific way. It needs to be adjusted to suit each individual part. The Parts surface needs to be a relatively rough and have an almost matte quality to it. Shiny smooth surfaces are not picked up well or at all by the algorithm which makes it use in an industrial application to check metallic parts very limited. It currently requires the use of a CNC machine to hold the camera which is a limitation for anyone who doesn't have access to one. A

computer controlled moving stage specifically for this would be a fairly easy thing to make. Then the image gathering process could be done much more quickly. Much like some of the fledgling first attempts at any other technical achievement, more research will need to be done before this is a viable method of determining part shape.

This method of 3D scanning shows promise and has some definite benefits. Only one camera is needed. This can definitely save on costs. It is reasonably accurate under certain conditions. Other vision systems of similar accuracy can cost many times as much. However, other vision systems can also offer many more capabilities. For a simple machine vision system, this process could easily be used to verify part accuracy. With some more work, this simple shape from focus operator could prove to be very useful in an industrial application. This endeavor to learn more about machine vision, focus measure operators, optics, and MATLAB was a success. This project has been a great learning experience and was a great opportunity to gain an understanding of optics and image processing.

## References

[1] S. Pertuz, D. Puig and M. A. Garcia, "Analysis of focus measure operators for shape-from-focus," *Pattern Recognition,* pp. 1415-1432, 2013.

[2] R. Minhas, A. A. Mohammed and Q. J. Wu, "Shape from focus using fast discrete curvelet transform," *Pattern Recognition,* pp. 839-853, 2011.

[3] S.-O. Shim and T.-S. Choi, "A novel iterative shape from focus algorithm based on combinatorial optimization," *Pattern Recognition,* pp. 3338-3347, 2010.

[4] I.-H. Lee, S.-O. Shim and T.-S. Choi, "Improving focus measurement via variable window shape on surface radiance distribution for 3D shape reconstruction," *Optics and Laser Engineering,* pp. 520-526, 2013.

[5] M. T. Mahmood and T.-S. Choi, "3D shape recovery from image focus using kernel regression in eigenspace," *Image and Vision Computing,* pp. 634-643, 2010.

[6] Y. Liu, G. Zhai, X. Liu, D. Zhao and W. Gao, "Quality assessment for real out-of-focus blurred images," *J. Vis. Commun. Image R.,* pp. 70-80, 2017.

[7] X. Zhang, H. Wu and Y. Ma, "A new auto focus measure based on medium frequency discrete cosine transform filtering and disct=rete cosine transform," *Applied and Computational Harmonic Analysis,* pp. 430-437, 2016.

[8] X. Xia, Y. Yao, J. Liang, S. Fang, Z. Yang and D. Cui, "Evaluation of focus measures for the autofocus of line scan cameras," *Optik,* pp. 7762-7775, 2016.

[9] B. Luthi, N. Thomas, S. Hviid and P. Rueffer, "An efficient autofocus algorithm for a visible microscope on a Mars lander," *Planetary and Space Science*, pp. 1258-1264, 2010.

[10] R. Chen and P. V. Beek, "Improving the accuracy and low-light performance of contrast-based autofocus using supervised machine learning," *Pattern Recognition Letters*, pp. 30-37, 2015.

[11] X. Sha, W. Li, X. Lv and Z. Li, "Research on auto focusing technology for micro vision system," *Optik*, pp. 226-233, 2017.

[12] H. Nanda and R. Cutler, "Practical calibrations for a Real-Time digital omnidirectional camera, technical report, technical sketches," *Computer Vision and Pattern Recognition*, 2001.

[13] S. Pertuz, D. Puig and M. A. Garcia, "Reliability measure for shape-from-focus," *Image and Vision Computing*, pp. 725-734, 2013.

[14] S.-S. Tung and W.-L. Hwang, "Multiple depth layers and all in focus image generations by blurring and deblurrign operations," *Pattern Recognition*, pp. 184-198, 2017.

[15] R. Rahmat, A. S. Malik, N. Kamel and H. Nisar, "3D shape from focus using LULU operators and discrete pulse transform in the presence of noise," *J. Vis. Commun. Image R.*, pp. 303-317, 2013.

[16] C.-J. Chen, K.-C. Huang, C.-L. CHang and M.-W. Hung, "The development of inclination measurement by using the imaging based autofocus method," *Physics Procedia*, pp. 43-48,

2011.

[17] O.-J. Kwon, S. Choi, D. Jang and H.-S. Pang, "All in focus imaging using average filter based relative focus measure," *Digital Signal Processing,* pp. 200-210, 2016.

[18] M. Nejati, S. Samavi, N. Karimi, S. R. Soroushmehr, S. Shirani, I. Roosta and K. Najarian, "Surface area based focus criterion for multi focus image fusion," *Information Fusion,* pp. 284-295, 2016.

[19] M. J. Amin and N. A. Riza, "Active depth from defocus system using coherent illuminiation and a no moving parts camera," *Optics Communications,* pp. 135-145, 2015.

[20] L. Zhao, L. Wang, Bi, S. Li, L. Yang and H. Zhang, "Structured sparsity driven autofocus algorithm fo rhigh resolution radar imagery," *Signal Processing,* pp. 376-388, 2016.

[21] Y. Zhang, X. Bai and T. Wang, "Boundary finding based multi focus image fusion through multi scale morphological focus measure," *Information Fusion,* pp. 81-101, 2016.

[22] D. Xu, "A unified approach to autofocus and alignment for pattern localization using hybrid weight Hausdorff distance," *Pattern Recognition Letters,* pp. 1747-1755, 2011.

[23] A. J. hunter, B. W. Drinkwater and P. D. Wilcox, "Autofocus of ultrasonic imagery for non destructive testing and evaluation of specimens with complicated geometries," *NDT&E International,* pp. 78-85, 2009.

[24] G. Bruls, "Exact formulas for a thin lens system with an arbitrary number of lenses," *Optik,* pp. 659-662, 2015.

[25] G.-h. Zong, M.-L. Sun, S.-s. Bi and D. Dong, "Research on Wavelet Based Autofocus Evaluation in Micro-vision," *Chinese Journal of Aeronautics,* vol. 19, no. 3, August 2006.

## APPENDIX A. MATLAB CODE EXAMPLE

# Thesis Code; Mitchel Wendland 2017

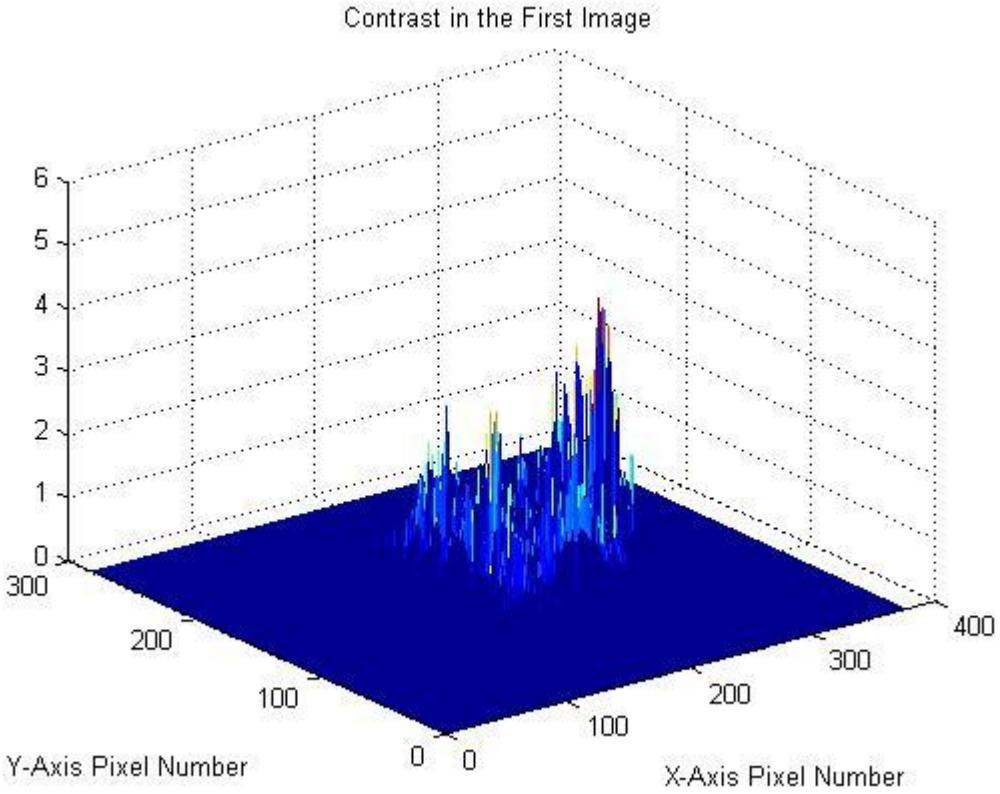
```
clear;
clc;
n=200; % number of images
N=n; % also number of images
s=.002; % step size in inches
so=.375; % size of the object in inches
xres=659; % long axis resolution
yres=494; % short axis resolution
fovx=degtorad(76.7); % field of view in degrees in the x or long axis
fovy=degtorad(57.4); % field of view in degrees in the y or short axis
pas=123; % pixels across the object when it is small in the picture
pal=166; % pixels across the object when the object is large in the image
theta=degtorad(radtodeg(fovx)/xres); % angular size of one pixel
(average) start=(so/2)/(tan(theta*pas/2)); % start height of the images
in inches
                                    % calculated using theta and the
                                    width at % a known distance
finish=start-N*s; % end of the step location will be less than start

% calculating the image width of the last or closest image
ihl=2*tan(fovx/2)*finish; % image width in inches of last
image iwl=2*tan(fovy/2)*finish; % image height in inches of
last image ih=2*tan(fovx/2)*start; % image width in inches
of first image iw=2*tan(fovy/2)*start; % image height in
inches of first image ws=(iw-iwl)/N; % image step size x or
long axis hs=(ih-ihl)/N; % image step size y or short axis

x3=atan(.5*iwl/start)/theta; % solving for where to crop

% location of the crop in pixels for the first image

X3=round(xres/2-x3); % left x coordinate for
crop X4=round(xres/2+x3); % right x coordinate
for crop

config=1; % contrast calculation 0 for plus 1 for 8 point
noise=1; % time saving filter 0 for off 1 for on (results in a less
       % accurate depth map, but cuts computation time.)
nf=.4; % noise reduction factor 1 keeps only the contrast that is
above
      % average value for the image.
filter=5; % number of times to filter the
data
f=.93; % pixel distortion size correction factor set to 1 for no
effect
       % should fade to 1 as it steps through the images
        % this value can be modified and played with in order to get
          the
        % best looking plots. 0.93 default

% image cropping algorithm. this sizes each successive image in the
  stack
% to the same realtive size
for k=1:n
    filename=[num2str(k) '.jpg'];
    I=imread(filename); % Load Image
    d=start-k*s; % distance from the reference to the image

    % location of crop in pixels from the center with correction
    F=(1-f)/N*k+f;
```

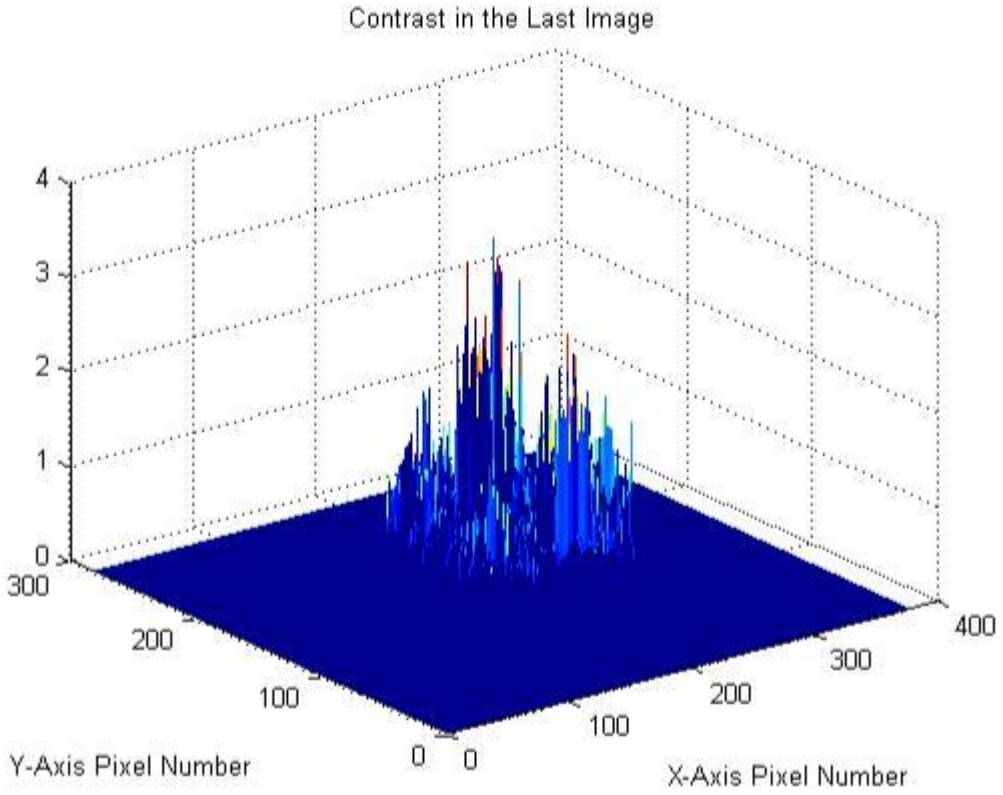
```
            y1=F*radtodeg(atan(.5*iwl/d))/(radtodeg(fovx)/xres); % crop
            location x1=F*radtodeg(atan(.5*ihl/d))/(radtodeg(fovx)/xres); %
            crop location

              % location of crop in pixels


            X1=round(xres/2-x1); % left x coordinate for crop in pixels
            X2=round(xres/2+x1); % right x coordinate for crop in pixels
            Y1=round(yres/2-y1); % upper y coordinate for crop in pixels
            Y2=round(yres/2+y1); % lower y coordinate for crop in pixels
            W=X2-X1; % width  of cropped image in pixels
            H=Y2-Y1; % height of cropped image in pixels
            rect=[X1 Y1 W H]; % assembling rect vector to feed to imcrop X
            and Y
                              % specify the location of the corner. W and H
                                are
                    %             the width and Height of the cropped
area. scale=(X4-X3)/(W); % assembling scale to feed to imresize so each
                               % image has the same number of
            pixels I=imcrop(I,rect); % crop the image to size
            specified in rect I=imresize(I,scale); % resize the
            image
            Ip=rgb2hsv(I); % convert image from RGB to HSV Value =
            brightness if k==round(N/2) % saving the color map from the
            middle image for the
                              % surface plot. This will be overlaid on the
                                plot,
                              % and is a good representation of what the part
                              % looks like in decent focus.
                Is=imread(filename);
                Is=imcrop(Is,rect); % crop the image to size specified
                in rect Is=imresize(Is,scale); % resize the image u=Is;

            end

            % Calculate contrast in the Images. either + or 8 point
              configuration.
            % 8 point seems to produce a slightly more accurate plot, but
              takes
            % significantly more time.
            Ca=mean(mean(Ip(:,:,3)));
            for j=2:(length(Ip(1,:,1))-1)
                for i=2:(length(Ip(:,1,1))-1)
                    if config==1
                        C(i,j,k)=abs(((Ip(i+1,j,3)+Ip(i-
                        1,j,3)+Ip(i,j+1,3)+...
                      Ip(i,j-1,3)+Ip(i+1,j-1,3)+Ip(i-1,j+1,3)+Ip(i+1,j+1,3)...
                            +Ip(i-1,j-1,3))-8*Ip(i,j,3))/Ip(i,j,3));
                                % 8 point contrast calculation
                    end
                    if config==0
                        C(i,j,k)=abs(((Ip(i+1,j,3)+Ip(i-
                        1,j,3)+Ip(i,j+1,3)+...
                            Ip(i,j-1,3))-4*Ip(i,j,3))/Ip(i,j,3));
                                % plus shaped contrast calculation
                    end
                    if noise==1
                        if abs(C(i,j,k))>Ca*nf  % this makes for nicer plots of
                                                % single image contrast by
                                                tossing
                                                % lower values
                    else
                        C(i,j,k)=0;
                        end
                    end
```

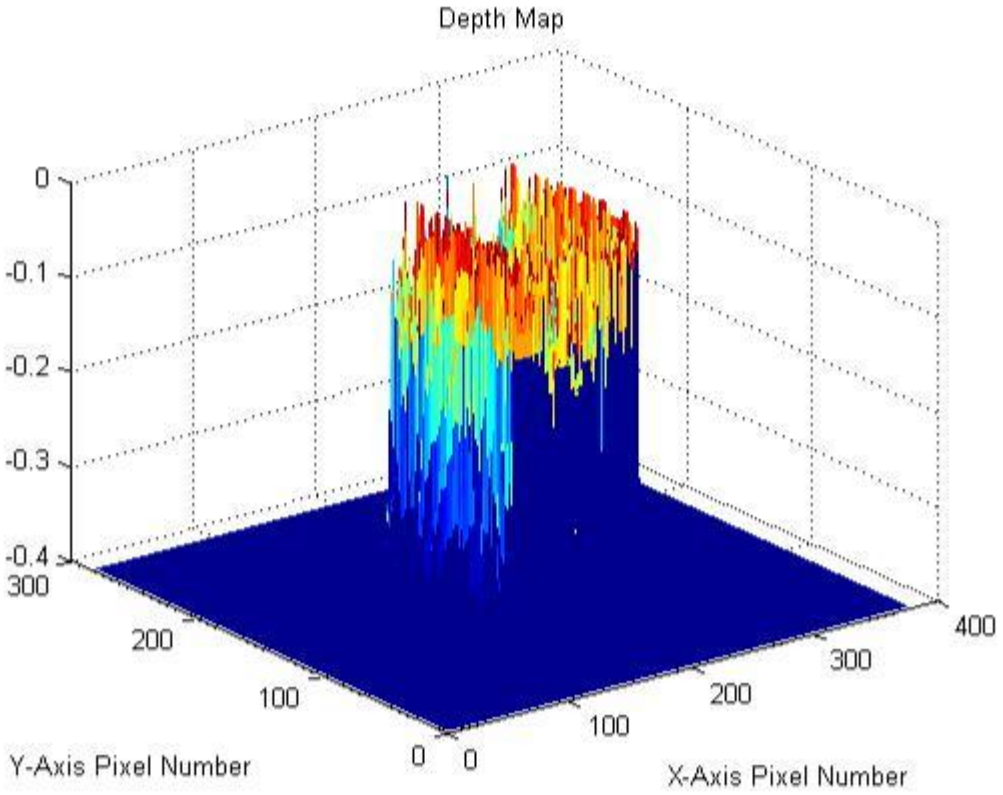
```
                end
            end
            contrast(k)=mean(mean(C(:,:,k)));
        end

        figure (1);
            surf(C(:,:,1),'EdgeColor','none');
            title('Contrast in the First Image');
                xlabel('X-Axis Pixel Number');
                ylabel('Y-Axis Pixel Number');
        figure (2);
            surf(C(:,:,N),'EdgeColor','none');
            title('Contrast in the Last Image');
                xlabel('X-Axis Pixel Number');
                ylabel('Y-Axis Pixel Number');

% comparing the contrast from one image to the next

Ch=zeros(length(C(:,1,1)),length(C(1,:,1)));


        for k=1:n-1
           for
                j=1:(length(C(1,:,1)
                )) for
                i=1:(length(C(:,1,1)
                ))
                  if abs(C(i,j,k+1)) >= abs(C(i,j,k));
                      if abs(C(i,j,k+1)) >=Ch(i,j)
                         model(i,j,1)=k*s;  %  gives  the  z  coordinate  of  a
                         particular
                                          % pixel when it is in best
                         focus Ch(i,j)=abs(C(i,j,k+1));
                         end
                    end
                end
            end
        end
        figure (3);
        surf(-1*model,'EdgeColor','none')
        title('Depth Map');
            xlabel('X-Axis Pixel Number');
            ylabel('Y-Axis Pixel Number');


        figure (4);
        surf(-1*model,u,'FaceColor','texturemap',...
            'EdgeColor','none',...
            'CDataMapping','direct')
            title('Middle Image Overlay on Depth
                Map'); xlabel('X-Axis Pixel
                Number'); ylabel('Y-Axis Pixel
                Number');

        F = .0205*ones(7); % using a convolution filter to smooth the data
        model1=model;
        model3=model;

%         edges which are ruined by the filter are removed and
%         the center of the plots can be more easily
seen. rem=1/10; % border width in a fraction or a
decimal aa=size(model1);
        AA=round(aa(1)*rem)
        ;
        AB=round(aa(1)*(1-
```

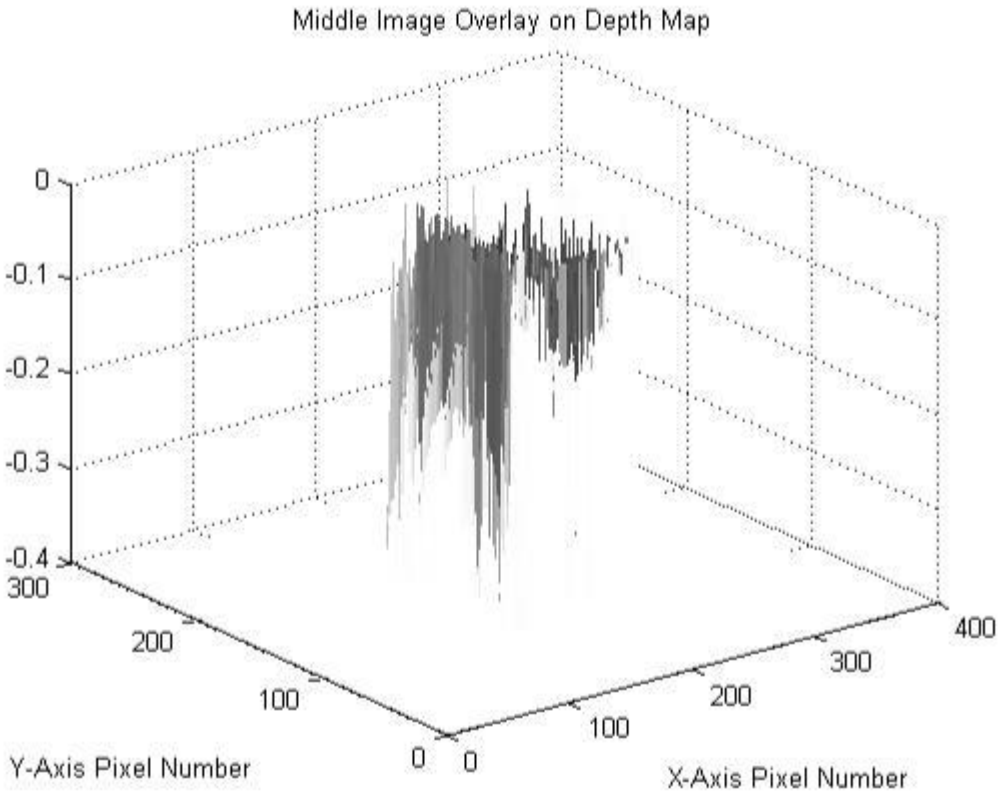
```
rem));
BA=round(aa(2)*rem)
;
BB=round(aa(2)*(1-
rem));

for v=1:filter
model1 = conv2(model1,F,'same');
end

figure (5);
surf(-1*model1(AA:AB,BA:BB),'EdgeColor','none')
    title('Smoothed Depth Map');
        xlabel('X-Axis Pixel Number');
        ylabel('Y-Axis Pixel Number');
        shading interp;

figure (6);
surf(-
1*model1(AA:AB,BA:BB),u(AA:AB,BA:BB,:),'FaceColor','texturemap',...
    'EdgeColor','none',...
    'CDataMapping','direct');
    title('Middle Image Overlay on Smoothed Depth
        Map'); xlabel('X-Axis Pixel Number');
        ylabel('Y-Axis Pixel Number');

figure (7);
contour(-1*model1(AA:AB,BA:BB),20);
    title('Contour Plot of Smoothed Depth Map');


        xlabel('X-Axis Pixel Number');
        ylabel('Y-Axis Pixel Number');

%         this part assigns an x and y value to each pixel so we can get a
rough
%         idea of the actual scale.
xoffset=round(length(model1(1,:))/2);
yoffset=round(length(model1(:,1))/2);
model2=model1;

for x=1:length(model2(:,1,1))
    X(x)=start*tan((x-
    xoffset)*theta)*1.29;
    model2(x,:,2)=X(x);
end

for y=1:length(model2(1,:,1))
    Y(y)=start*tan((y-
    yoffset)*theta)*1.32;
    model2(:,y,3)=Y(y);
end

figure (8);
surf(model2(AA:AB,BA:BB,2),model2(AA:AB,BA:BB,3),-
1*model2(AA:AB,BA:BB,1));
    title('3d Model of Smoothed Data');
    xlabel('X-location in Inches');
    ylabel('Y-Location in Inches');
    colormap('hot');

figure (9);
surf(model2(AA:AB,BA:BB,2),model2(AA:AB,BA:BB,3),-
1*model2(AA:AB,BA:BB,1)...
    ,u(AA:AB,BA:BB,:),'FaceColor','texturemap',...
```

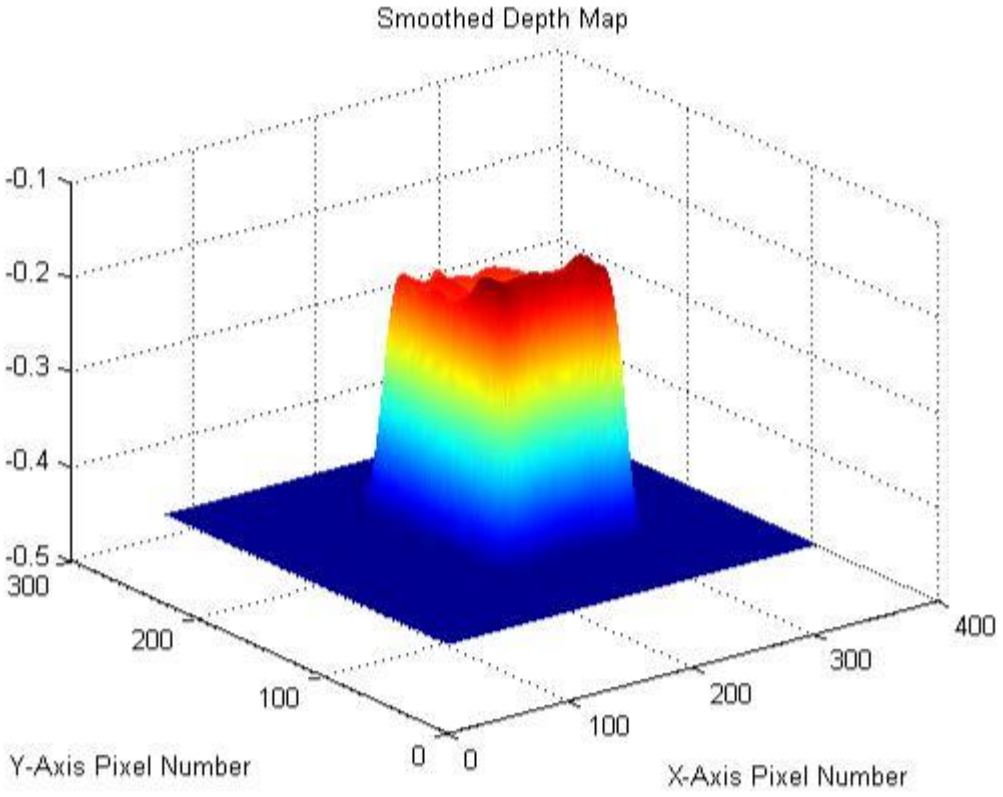
```
    'EdgeColor','none','CDataMapping','direct');
    title('3d Model of Smoothed Data with Middle Image
    Overlay'); xlabel('X-location in Inches'); ylabel('Y-
    Location in Inches');

figure (10);
plot(1:n,contrast,'blue');
    title('Mean Contrast Value vs. Image Number ');
    xlabel('Image Number');
    ylabel('Mean Contrast Value');

figure (11);
contour3(model2(AA:AB,BA:BB,2),model2(AA:AB,BA:BB,3),...
    -1*model2(AA:AB,BA:BB,1),20);
hold on
surf(model2(AA:AB,BA:BB,2),model2(AA:AB,BA:BB,3),-
1*model2(AA:AB,BA:BB,1)...
    ,u(AA:AB,BA:BB,:),'FaceColor','texturemap',...
    'EdgeColor','none','CDataMapping','direct');
    title('3d Model of Smoothed Data with Middle Image
    Overlay'); xlabel('X-location in Inches'); ylabel('Y-
    Location in Inches');
hold off

figure (12);
contour3(model2(AA:AB,BA:BB,2),model2(AA:AB,BA:BB,3),...
    -1*model2(AA:AB,BA:BB,1),20);
hold on
surf(model2(AA:AB,BA:BB,2),model2(AA:AB,BA:BB,3),-
1*model2(AA:AB,BA:BB,1)...
    ,u(AA:AB,BA:BB,:),'FaceColor','texturemap',...
    'EdgeColor','none','CDataMapping','dire
    ct'); colormap('hsv');
    title('3d Model of Smoothed Data with Middle Image
    Overlay'); xlabel('X-location in Inches'); ylabel('Y-
    Location in Inches');
hold off


figure (13);
contour3(model2(AA:AB,BA:BB,2),model2(AA:AB,BA:BB,3),...
    -1*model2(AA:AB,BA:BB,1),20);
```

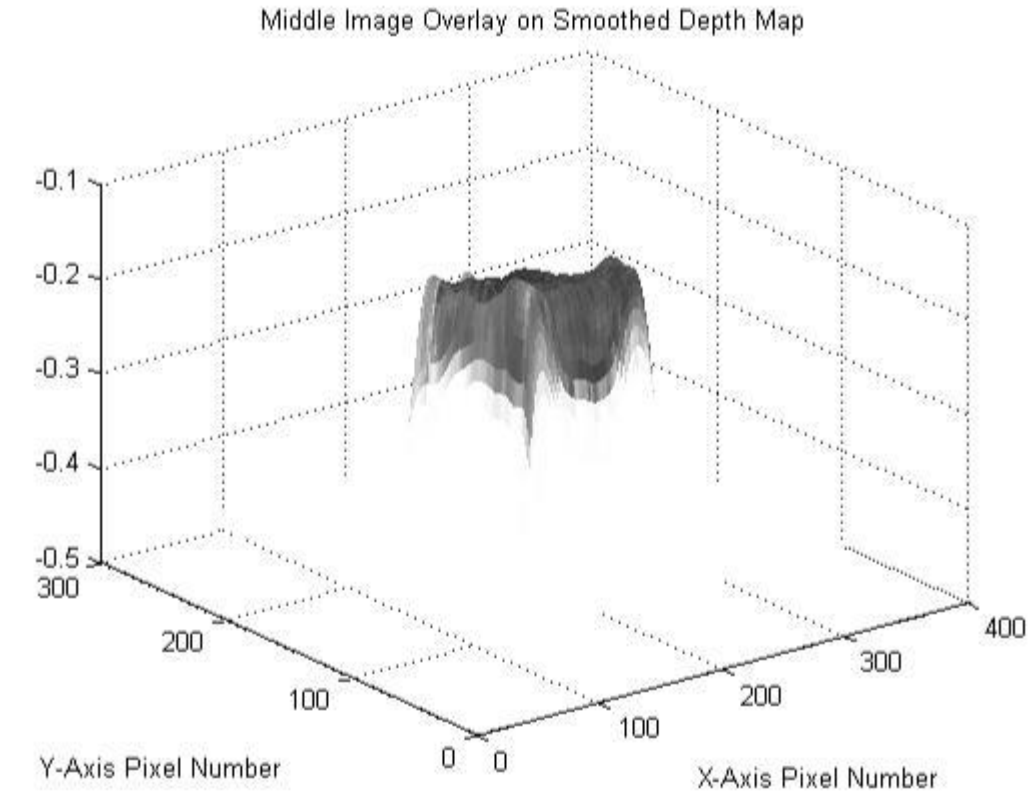

Contrast in the First Image
6
5
4
3
2
1
0
300
200
100
0
Y-Axis Pixel Number
0
100
200
300
400
X-Axis Pixel Number

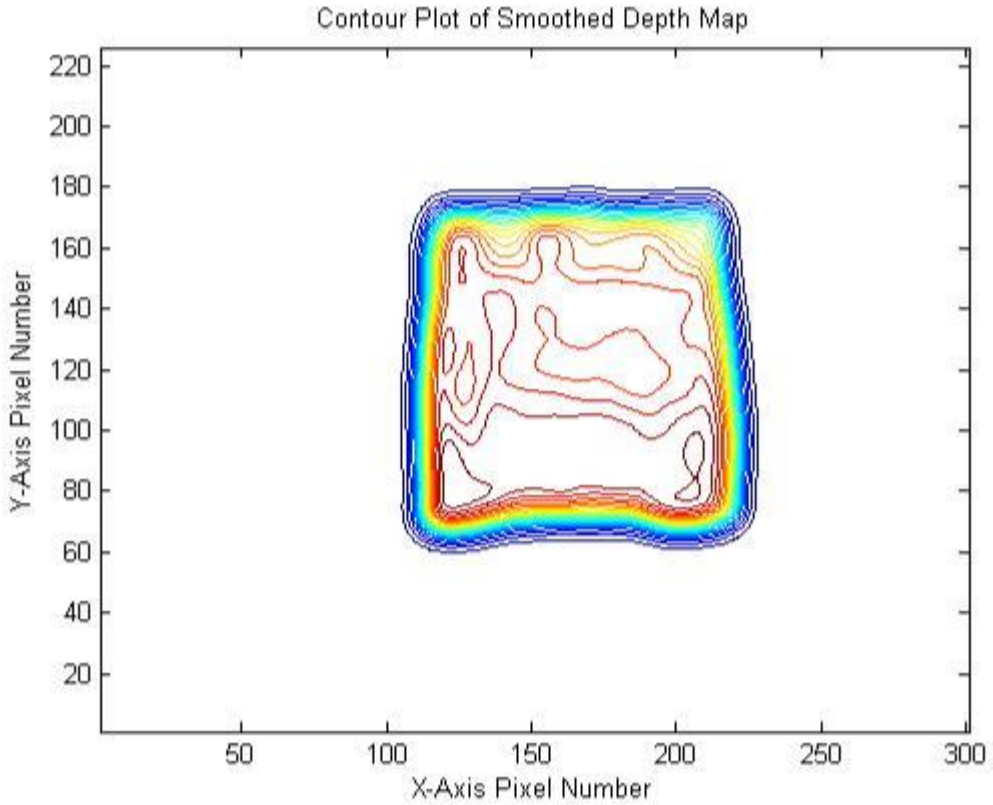

Contrast in the Last Image
4
3
2
1
0
300
200
100
0
0
100
200
300
400
Y-Axis Pixel Number
X-Axis Pixel Number

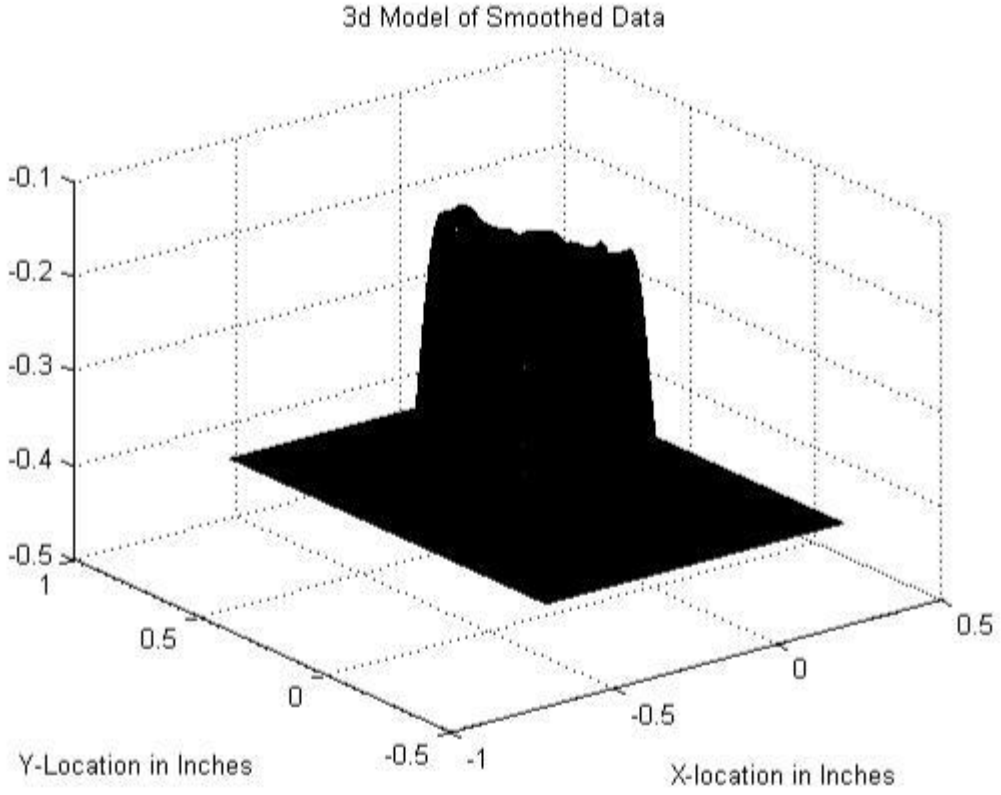

Depth Map
0
-0.1
-0.2
-0.3
-0.4
300
200
100
0
0
100
200
300
400
Y-Axis Pixel Number
X-Axis Pixel Number

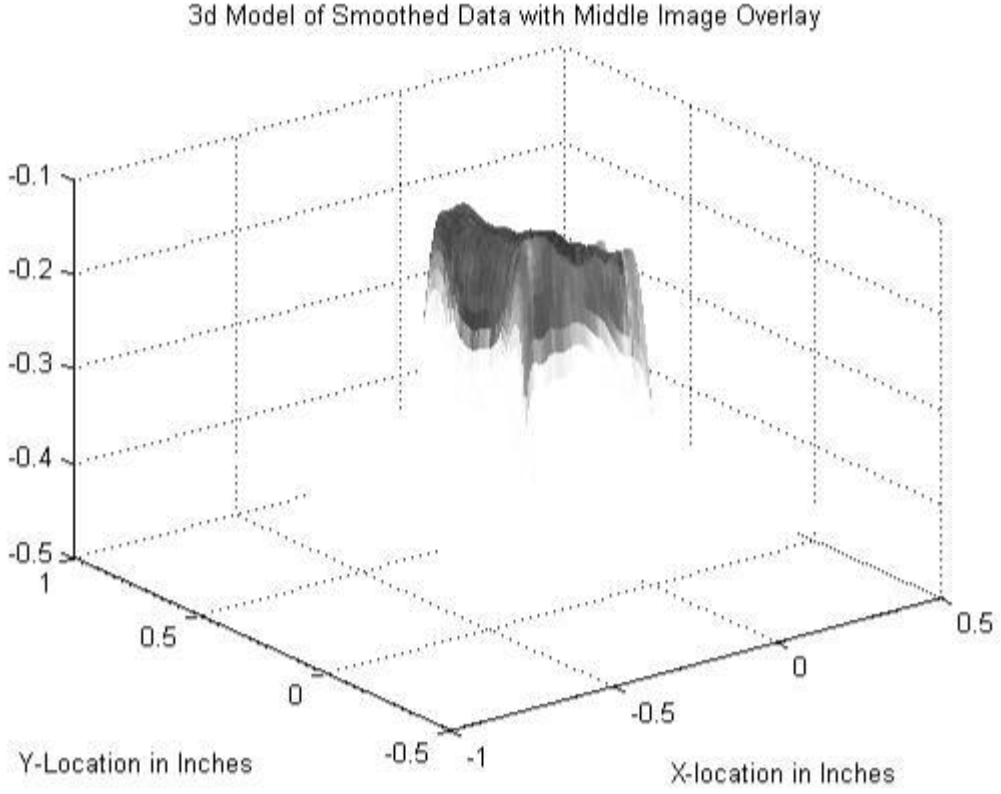

Middle Image Overlay on Depth Map
0
-0.1
-0.2
-0.3
-0.4
300
200
100
0
Y-Axis Pixel Number
0
100
200
300
400
X-Axis Pixel Number

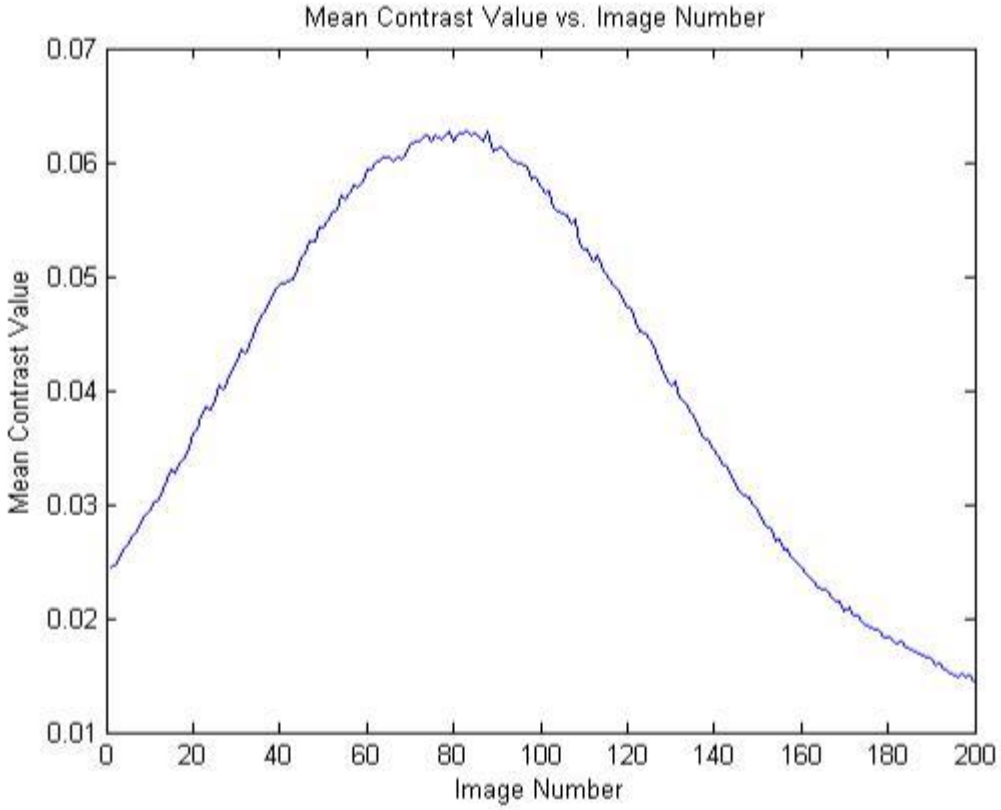

Smoothed Depth Map
-0.1
-0.2
-0.3
-0.4
-0.5
300
200
100
0
Y-Axis Pixel Number
0
100
200
300
400
X-Axis Pixel Number

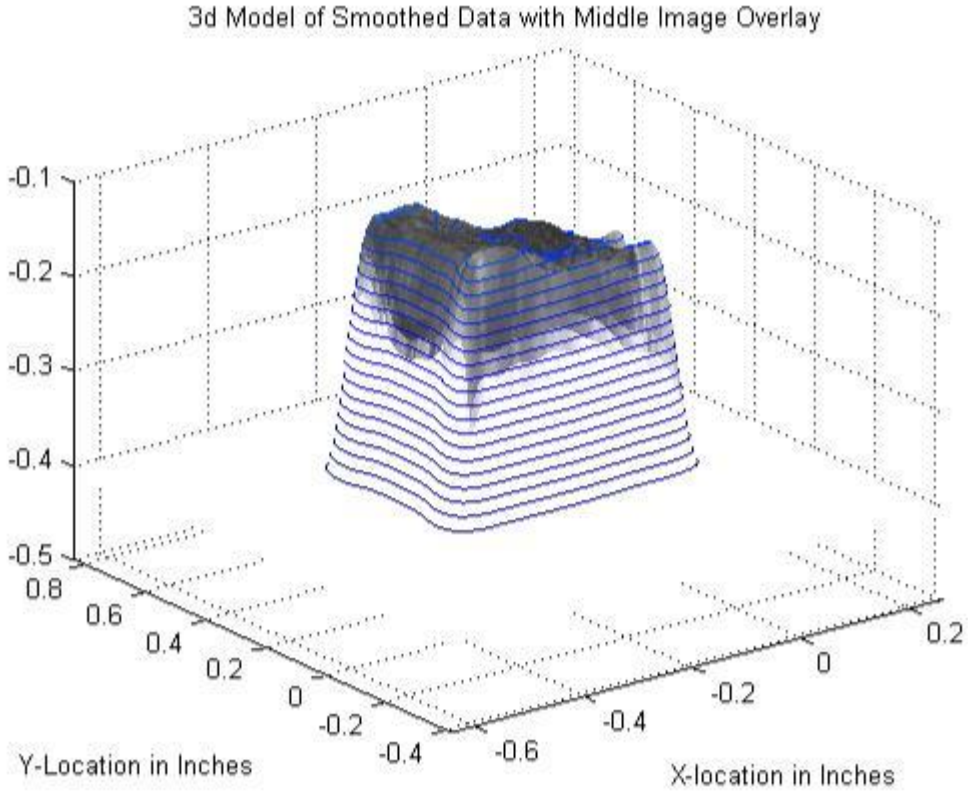

Middle Image Overlay on Smoothed Depth Map
-0.1
-0.2
-0.3
-0.4
-0.5
300
200
100
0
Y-Axis Pixel Number
0
100
200
300
400
X-Axis Pixel Number

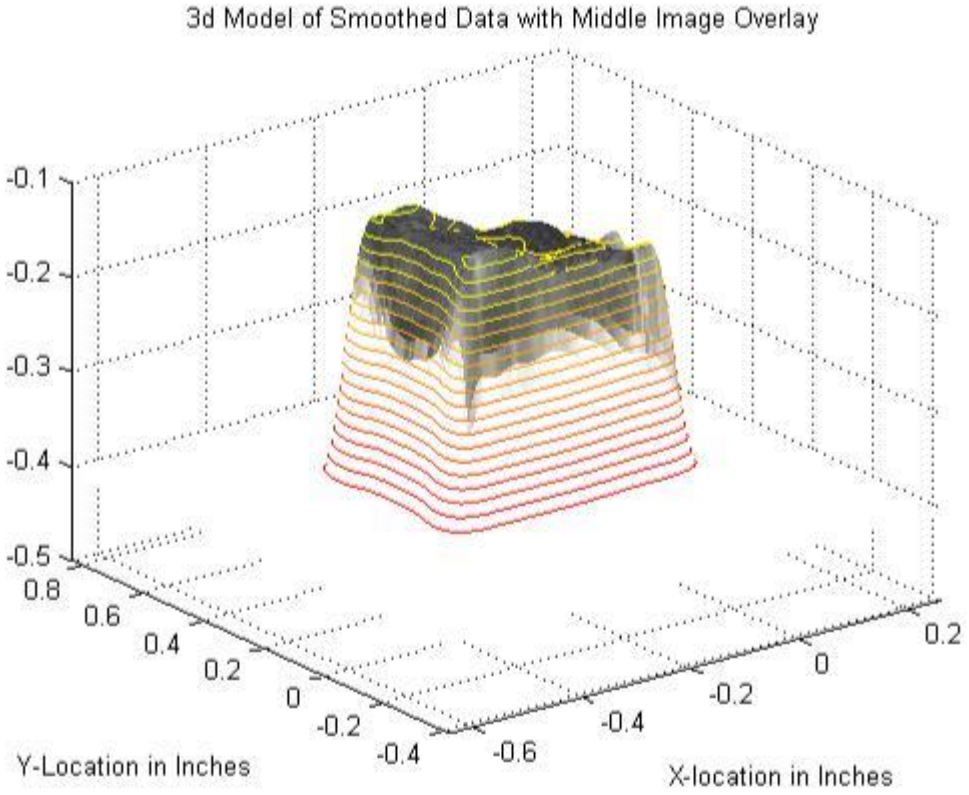

Contour Plot of Smoothed Depth Map
220
200
180
160
140
120
100
80
60
40
20
Y-Axis Pixel Number
50
100
150
200
250
300
X-Axis Pixel Number

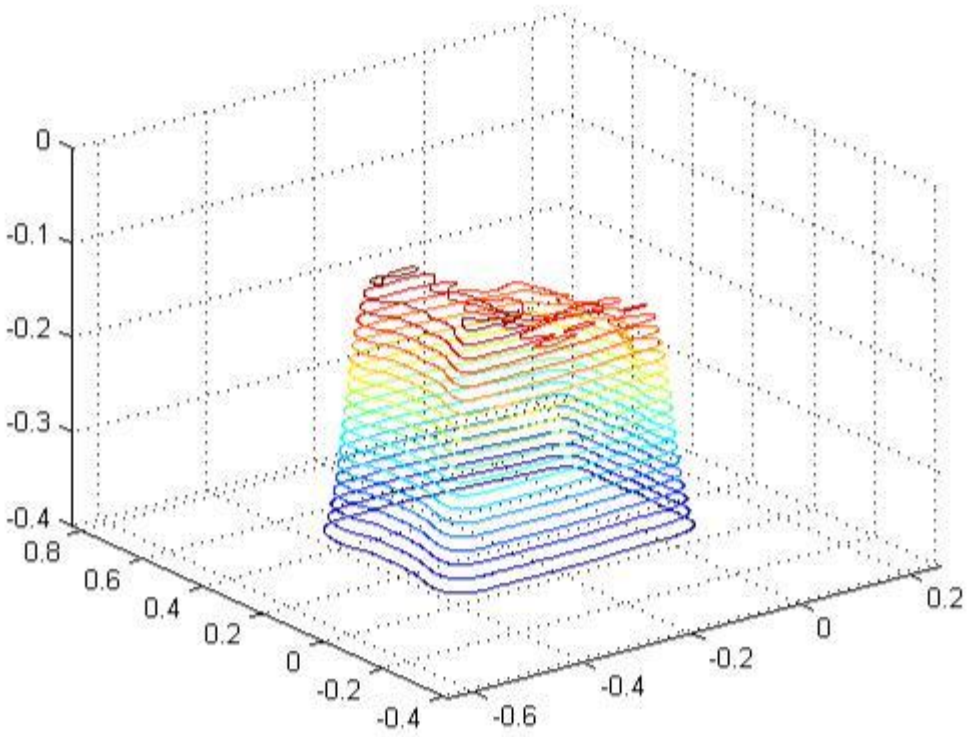

3d Model of Smoothed Data
-0.1
-0.2
-0.3
-0.4
-0.5
1
0.5
0
-0.5
Y-Location in Inches
-1
-0.5
0
0.5
X-location in Inches


3d Model of Smoothed Data with Middle Image Overlay
-0.1
-0.2
-0.3
-0.4
-0.5
1
0.5
0
-0.5
Y-Location in Inches
-1
-0.5
0
0.5
X-location in Inches


Mean Contrast Value vs. Image Number
Mean Contrast Value
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
20
40
60
80
100
120
140
160
180
200
Image Number


3d Model of Smoothed Data with Middle Image Overlay
-0.1
-0.2
-0.3
-0.4
-0.5
0.8
0.6
0.4
0.2
0
-0.2
-0.4
Y-Location in Inches
-0.6
-0.4
-0.2
0
0.2
X-location in Inches


3d Model of Smoothed Data with Middle Image Overlay
-0.1
-0.2
-0.3
-0.4
-0.5
0.8
0.6
0.4
0.2
0
-0.2
-0.4
Y-Location in Inches
-0.6
-0.4
-0.2
0
0.2
X-location in Inches


0
-0.1
-0.2
-0.3
-0.4
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.4
-0.2
0
0.2